AD-A258 717

CUMENTATION PAGE

Form Approved
OMB No. 0704-0188

ation is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources
pleting and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this
educing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson
2, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1992	THESIS

4. TITLE AND SUBTITLE
A Framework for Corporate Implementation of Object-Oriented Software Development

5. FUNDING NUMBERS

6. AUTHOR(S)
James Mark McVay, Captain

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
AFIT Student Attending: Colorado State University

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/CI/CIA-92-109

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFIT/CI
Wright-Patterson AFB OH 45433-6583

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for Public Release IAW 190-1
Distributed Unlimited
ERNEST A. HAYGOOD, Captain, USAF
Executive Officer

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

DTIC
ELECTE
DEC 0 8 1992
S B D

92-31015
208pp

14. SUBJECT TERMS	15. NUMBER OF PAGES 195
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

A FRAMEWORK FOR CORPORATE IMPLEMENTATION OF OBJECT-ORIENTED SOFTWARE DEVELOPMENT

by

JAMES MARK McVAY

Captain, USAF

1992

195 pages

Master of Science

University of Colorado

ABSTRACT

The object paradigm is praised by many software engineers as a solution for managing today's complex software development. Applying the paradigm through object-oriented analysis, object-oriented design, and object-oriented programming promises many benefits. Code is easier to reuse. Transitions across the development phases are more understandable. System development costs and time are reduced.

Reaping the benefits of object-oriented software development (OOSD) comes at the expense of major changes. Before fully committing to these changes a company must answer three basic questions:

1) Is the object paradigm useful to software development?

2) If so, how can it be applied to reap the most benefits?

3) What are the impacts on organizational dynamics?

The difficulty with these questions is that none can be answered without experience. In order for the object paradigm to be judged, it must be tested.

This thesis proposes a framework which provides guidance in establishing an adequate test environment. The framework consists of ten components. Each component addresses a specific area of the software development environment. Each component is constructed to highlight the implementation factors required by OOSD.

Following the framework is a case study of one company which

developed a system using object-oriented techniques. This case study is based on the framework and was designed to test the viability of the framework components.

The thesis serves as a beginning for other research necessary to support or adjust the framework in its goal of establishing a proper OOSD test environment.

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist Avail and/or Special
A-1

A FRAMEWORK FOR CORPORATE IMPLEMENTATION OF OBJECT-ORIENTED SOFTWARE DEVELOPMENT

by

JAMES MARK McVAY

B.S., Auburn University, 1983

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirement for the degree of
Master of Science
School of Business
1992

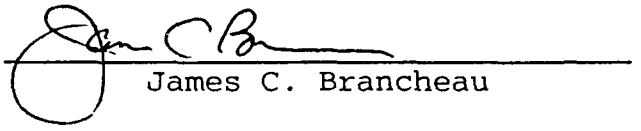
This thesis for the Master of Science degree by

James Mark McVay

has been approved for the

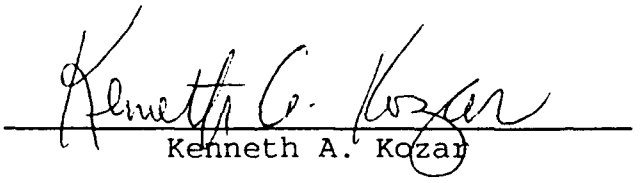
School of Business

by



James C. Brancheau

David A. Carlson



Kenneth A. Kozar

Date 7/16/92

McVay, James Mark (M.S., Management Science and Information Systems)

A Framework for Corporate Implementation of Object-Oriented Software Development

Thesis directed by Assistant Professor James C. Brancheau

The object paradigm is praised by many software engineers as a solution for managing today's complex software development. Its scope covers all phases of a software development life cycle. Applying the paradigm through object-oriented analysis, object-oriented design, and object-oriented programming promises many benefits. Code is easier to reuse. Transitions across the development phases are more understandable. System development costs and time are reduced to yield higher returns on investment.

Reaping the benefits of object-oriented software development (OOSD) comes at the expense of major changes. Traditional software developers must convert to a technology conceptually opposed to past practices. The initial costs for converting to OOSD are quite high. New training, tools, and management practices are required. Before fully committing to these changes a company must answer three basic questions:

1) Is the object paradigm useful to software development?

2) If so, how can it be applied to reap the most benefits?

3) What are the impacts on organizational dynamics?

The difficulty with these questions is that none can be answered without experience. In order for the object paradigm to be judged, it must be tested.

This thesis proposes a framework which provides guidance in establishing an adequate test environment. The framework consists of ten components. Each component addresses a specific area of the software development environment. Each component is constructed to highlight the implementation factors required by OOSD.

Following the framework is a case study of one company which developed a system using object-oriented techniques. This case study is based on the framework and was designed to test the viability of the framework components.

While one case study is not sufficient to validate the framework, two components were heavily supported by the study. The thesis serves as a beginning for other research necessary to support or adjust the framework in its goal of establishing a proper OOSD test environment.

CONTENTS

CHAPTER I

INTRODUCTION

Commercial software development began in 1951 when UNIVAC I was delivered to the U.S. Bureau of Census.[1] Today's hardware eclipses the abilities of yesterday's and continues to advance in power. Its capacity demands software at an ever increasing rate. As a result, software systems continue to grow in complexity while developers search for ways to manage them. The accumulation of older programming languages and methods to apply them often frustrates this search. New development techniques find their way out of research, each promising it is up to today's challenge. With so many different development approaches, managers may question which choice is the best solution.

The object paradigm is one approach touted by many software engineers as a solution for today's complex development environments. Its application can reach across all phases of the software life cycle. It offers benefits such as development cost and time reductions. It promises to create more understandable documentation and code. Formerly rough transitions through the life cycle phases are supposedly smoothed. The benefits come

at a price, though. This price is organizational change.

Moad observes, "as object-oriented technologies move into IS organizations, there is growing recognition that . . . major changes in programmer training, development styles, and tools" are required.[2] Herzog warns, however, "management must appreciate the impact of change evolves over time. . . . Too frequently the change is not allowed to take root . . . and realize its full potential."[3] The change expected by the object paradigm requires that organizations compile sufficient information to determine if the change is worth the cost.

Problem Statement

Organizations considering the object paradigm are faced with three fundamental questions:

1) Is the object paradigm useful to software development?

2) If so, how can it be applied to reap the most benefits?

3) What are the impacts on organizational dynamics?

The difficulty with these questions is that none can be answered without experience. In order for the object paradigm to be judged, it must be tested. The problem is to establish an adequate environment to properly test the paradigm.

Thesis Purpose

The purpose of this thesis is to propose an appropriate environment under which object-oriented software development (OOSD) can be adequately tested. Firms with a valid test environment can find their own answers to the three fundamental questions.

The thesis will examine the foundations of the object paradigm and the reasons for its consideration. It will examine literature for current software management practices and their applicability to OOSD.

Method

This thesis proposes a framework which is designed to help establish an appropriate test climate. Ten components form the framework. Each component addresses a specific area of the software development environment. Each component is constructed to highlight the implementation factors required by the technology.

The thesis presents the definition of an object, the principles of the object-oriented approach, and the OOSD process. This information is needed to understand the relevance of the framework to OOSD.

The thesis also presents a case study based on the framework. The case study was designed to test the viability of the framework components. A company with a large OOSD project volunteered to participate in this

research. Initial contact with this company suggested the project was a success story. Detailed study revealed this was not necessarily true. Serious problems hindered the process and caused a late end product. Thirty-four pages of interview data reveal an intriguing story.

One case study is insufficient to adequately support or disprove the framework. However, the case study does suggest the framework's positions on vendor training and an appropriate software project are sound. Inadequate training and pioneering OOSD with a critical project contributed heavily to the above problems.

Thesis Organization

Chapter II begins by explaining the discipline and goals of software engineering. This is to show the reader why OOSD appears to be a viable software development technology. The definition of an object follows as does a synthesis of object-oriented principles.

Chapter III describes the application of OOSD through an appropriate life cycle. The phases of the life cycle are explained along with the respective benefits of each.

Chapter IV presents the framework's construction. Each component is defined via its respective attributes and operations. The thesis finds support for the components in numerous literature sources. Some attributes

and operations derive from the technology requirements.

Chapter V presents how the research was conducted. It covers the chosen research method, the process of contacting companies, how participants were selected, rationale of the question set design, and the goal of the research process.

Chapter VI analyses the data found in Appendix B. Interviewee answers are compared and contrasted. A summary of key findings closes the chapter.

Chapter VII concludes the thesis with closing discussions. The prognosis for OOSD diffusion in the studied firm is discussed along with potential areas for future research.

Notes - Chapter I

[1] Anthony Ralston and Edwin D. Reilly, Jr., eds. Encyclopedia of Computer Science and Engineering (New York: Van Nostrand Reinhold Company, 1983), s.v. "UNIVAC I" by Michael M. Maynard.

[2] Jeff Moad, "Cultural Barriers Slow Reusability," Datamation, 15 November 1989, 87.

[3] John P. Herzog, "People: The Critical Factor In Managing Change," Journal of Systems Management, March 1991, 8.

CHAPTER II

THE OBJECT PARADIGM

Using object-oriented concepts is seen by many as a new way to develop software; however, the idea has been around for many years. Most software professionals view the development of SIMULA 67 in the late sixties as the beginning of the object paradigm.[1][2] Ten Dyke and Kunz argue, however, that the 1957 Minuteman missile project used "primitive object-oriented techniques."[3] The idea that the object paradigm is new stems from its only recent acceptance as a means to develop software. Coad and Yourdon say this passing of time allowed the object paradigm to mature to a point where it is now a viable approach.[4]

The use of object-oriented techniques appears to hold great promise in the management of increasingly complex software systems. However, software professionals in the midst of making decisions based on this promise may find themselves confused by the extravagances of claims made in the media. One article will tell how object-oriented approaches will make software development simple.[5] Another will propose that using objects is a "silver bullet" in the battle over software prob-

lems.[6] Others may take the opposite position and argue the object paradigm is overrated and just will not match the claims made about it. This thesis takes the position that the promises of the object paradigm may be attainable, but not automatically so. While a technology may have many benefits, it takes humans to realize these benefits. Humans must create a proper environment to use the object paradigm before its promises can be expected to be fulfilled. The framework proposed by this thesis seeks to provide guidance in building this proper environment.

In order to understand how the object paradigm aids software development, one must first understand the discipline of software engineering. Object-oriented techniques are promising means to meet the objectives of software engineering.

Software Engineering

The complexity of modern software requires a disciplined approach to its development and management. The title "software engineering" is a broad term encompassing most of the specific activities involved in developing software. Though its scope is broad, software engineering does demand definite actions of software professionals. Its definition hints at these actions.

Boehm combines the separate dictionary meanings of

software and engineering to give us the following definition:

> Software engineering is the application of science and mathematics by which the capabilities of computer equipment are made useful to man via computer programs, procedures, and associated documentation.[7]

Boehm goes on to emphasize two key points of this definition. He insists that software be "useful to man" and that software professionals must recognize software development encompasses more than just the computer programs.[8]

The discipline of software engineering does not identify specific means to accomplish the requirements of the above definition (this is left to entrepreneurs as they develop marketable methods), but the discipline does specify **what** is necessary to fulfill the definition. Discipline requirements are revealed by examining Boehm's software engineering goal structure and Booch's goals of software engineering.

Goals of Software Engineering

Using Boehm's software engineering goal structure, this thesis will focus on his subgoal of achieving a successful software product. In the context of human relations, he says the software product should be easy to use, should satisfy an actual human need, and should enhance human potential. In the context of actual program engineering, he says the software product's requir-

ements should be precisely stated; the end-product should correctly match these requirements; the end-product should be adaptable to changing requirements and/or changing environments.[9]

Booch cites the work of Ross, Goodenough, and Irvine[10] in identifying four goals of software engineering: modifiability, efficiency, reliability, and understandability. He explains modifiability as "controlled change . . . without increasing the complexity of the original system."[11] This corresponds to Boehm's goal of adaptability. Efficiency is defined as "using the set of available resources in an optimal manner."[12] Booch identifies these resources as time and space within the context of hardware components. This thesis takes the view that these resources are too narrowly defined. Time and space are applicable in many contexts, especially in the context of human resources. This thesis expands the efficiency goal to include human time as a resource to be optimally used.

In setting reliability as a goal, Booch acknowledges no system is perfect. However, he expects "a reliable system would degrade gracefully without causing any dangerous side effects."[13] Booch identifies understandability as the most important goal of software engineering.[14] Without a clear understanding of what is to be done, the final product will not be easy to

use and will have no assurance of satisfying a need. It cannot be precisely specified or be tested effectively for correctness.

To be understandable, Booch says a system "must directly reflect our natural view of the world."[15] In other words the end-product should use processes and terminology consistent with how a human would interpret and perform a task. Booch restricts his explanation of understandability to the contexts of design and programming. This thesis takes the view that understandability should cloak not only computer programs but all procedures and associated documentation needed to build systems.

Combining Boehm's and Booch's goals results in the following composite list of software engineering goals:

1. Systems which are easy to use.
2. Systems which satisfy human needs and enhance human achievement.
3. Requirements which are clear and precisely stated.
4. Systems which are easily modified or adapted to changing requirements.
5. Efficiency of computer and human resources.
6. Systems which are reliable.
7. Understandability across all system development processes.

Applied in a disciplined manner, the object paradigm can aid tremendously in obtaining these goals. Before

examining the means of applying the object paradigm, though, a study of object fundamentals is needed.

Object Fundamentals

A quick review of the object-oriented material in print today reveals a great deal of confusion. Many authors use different terminology to define the expression "object-oriented." Though a standard consensus on defining the object paradigm is unavailable, a usable definition is attainable through careful study. This thesis extracts the similarities of the differing views and presents a definition of the term object and the general elements composing the term object-oriented.

Object Defined

Defining the term object is the easiest task. Most writers offer very similar meanings. Coad and Yourdon define an object as:

> An abstraction of something in a problem domain, reflecting the capabilities of a system to keep information about it, interact with it, or both; an encapsulation of attribute values and their exclusive services.[16]

Booch says:

> An object has state, behavior, and identity; the structure and behavior of similar objects are defined in their common class.[17]

Wirfs-Brock and Johnson offer:

> An object embodies an abstraction. It provides services . . . an object is not just a collection of

> data. The services are computations that are appropriate to the abstraction.[18]

Coad and Yourdon's information and attribute values, Booch's state, and Wirfs-Brock and Johnson's implication of a collection of data all point to the idea that objects possess data. The use of the term services in the first and third definition and of the term behavior in the second point to the idea that objects possess operations as well. Therefore, a general definition is: an object is a structure which contains both data and operations.[1]

The benefit of having a software construct as defined above is the overcoming of weaknesses of traditional software development techniques. Traditional techniques separate system development into functional modeling and data modeling. While successful at times in the past, this division creates problems as modern systems become more complex. Coad and Yourdon give two examples where software developers were divided into functional teams and data teams. This separation induced conflict and resulted in systems which were less than ideal.[19] Sommerville points out that "the role of any computer system is to model the real-world."[20]

Separating data structure from functionality is not a characteristic of real-world entities. As noted in

---

[1] This shorthand definition is commonly used in software engineering circles

literature, real-world entities have attributes (data) and operations (functions).[21] Humans are short, tall, blond, brunette, etc. They also drive cars, play baseball, are mugged, go to graduate school, etc. Objects give software developers the ability to map all these human aspects into one structure which reflects its real-world counterpart. Remember, this ability to map the real-world is key to Booch's requirement of an understandable system.

Object-Oriented Elements

Booch says object-oriented incorporates four primary elements: abstraction, encapsulation, modularity, and hierarchy.[22]

Booch defines abstraction in terms of levels of detail. He says, "a good abstraction . . . emphasizes details that are significant . . . and suppresses details that are . . . immaterial" for a given level.[23] Put in the context of objects, his definition is as follows:

> An abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer.[24]

He defines encapsulation as a complement of abstraction. He views abstraction as the "outside view" of an object, whereas encapsulation is the inside view or implementation of the object. He concisely defines

encapsulation as:

> A process of hiding all of the details of an object that do not contribute to its essential characteristics.[25]

Booch further explains these hidden details are known to all processes within the object's boundaries but are unknown to objects on the outside.

Booch defines modularity as:

> The property of a system that has been decomposed into a set of cohesive and loosely coupled modules.[26]

He quotes Britton and Parnas in stating, "the overall goal of the decomposition into modules is the reduction of software cost by allowing modules to be designed and revised independently."[27] He goes on to explain how modularity reduces system complexity by designing and coding smaller, more easily understood components. If done properly, modularity allows enhancements or changes to be made with little or no adverse effects to other components.

Booch defines hierarchy as simply "a ranking or ordering of abstractions."[28] He explains hierarchy in terms of inheritance or the obtaining of characteristics defined in one or more classes in a given hierarchy.

Korson and McGregor, Coad and Yourdon, and Ten Dyke and Kunz all include Peter Wegner's object-oriented elements with additions of their own. As Ten Dyke and Kunz point out, Wegner focused on languages and said for

a language to be considered object-oriented it must support objects, classes, and inheritance.[29] Ten Dyke and Kunz added dynamic binding and encapsulation as needed properties. Korson and McGregor add to Wegner the concepts of polymorphism and dynamic binding.[30] Coad and Yourdon add the principle of communication with messages to Wegner as the properties necessary for defining object-oriented.[31]

All three writings view objects, more or less, as defined above: a structure which contains data and operations.

Coad and Yourdon's definition of classes sums up the other two views:

> A description of one or more objects with a uniform set of attributes and services, including a description of how to create new objects in the class.[32]

The key points are: classes describe characteristics and objects are created from classes.

Each view inheritance much as Booch defines hierarchy. Ten Dyke and Kunz offer a succinct definition which encompasses the other two's discussions:

> Inheritance is a property of classes that allows them to share resources. Classes may be arranged in a hierarchy from most general to most specific. Classes lower in the hierarchy may inherit methods and attributes from classes above.[33]

Dynamic binding addresses the idea of waiting until run-time to allocate memory to a unit of code. In an object-oriented context, dynamic binding is usually

associated with polymorphism.

Polymorphism is a difficult concept to many traditional software developers but is a powerful, time saving tool when used correctly. Stefik and Bobrow define polymorphism as:

> The capability for different classes of objects to respond to exactly the same protocols. Protocols enable a program to treat uniformly objects that arise from different classes. A critical feature is that even when the same message is sent from the same place in code, it can invoke different methods.[34]

Understanding this definition requires an understanding of all previously defined concepts along with two others: methods and messages. Methods are synonymous with the operations associated with an object. An object's methods define what an object can do and what can be done to the object. For one object to cause another object's method to execute, it must communicate a request via a message. The above definition is saying that an object can send a message and, depending on the point in time, this message can be applied to different objects. If the addressed objects have a method matching the message, then that method is invoked. Because the object issuing the message is not required to know the class of the object being sent the message, the actual method code which is invoked is not known until run-time.

Booch illustrates the power of polymorphism with a language which does not support it. He shows how, with

Pascal, a class hierarchy cannot be created for variant but similar data objects. To invoke the correct methods, a programmer would first have to write code to decide which object is to be addressed and therefore which method to call. Any additions to the data would require error-prone changes to the already large testing code.[35] Polymorphism reduces the amount of coding required as long as the protocols are written correctly.

Coad and Yourdon's discussion on communicating with messages is limited due to its minimum application in analysis. Their views are encompassed by the more comprehensive discussion on polymorphism above.

Ten Dyke and Kunz's definition of encapsulation is encompassed by that of Booch's.

Although the above authors use differing terminologies and omit concepts which others include, the definite similarities and sound arguments allow this thesis to identify the following as the essential object-oriented elements:

1. Use of objects.
2. Abstraction.
3. Encapsulation.
4. Inheritance.
5. Polymorphism.

The use of objects as essential is obvious and precludes defending.

The thesis chooses the term abstraction over classes due to its encompassing the meaning of classes and its further use in managing the complexity of human thought. Abstraction is the beginning point in finding a software solution using objects.

This thesis chose encapsulation primarily by consensus. Nearly all authors touched on this concept in one way or another. Some may use the term information hiding. Korson and McGregor discussed the need for information hiding under the heading of classes. They later show information hiding is synonymous with encapsulation.[36] The principles associated with encapsulation are vital in incorporating the object characteristics identified by abstraction.

Inheritance was a common term among all authors. Booch listed hierarchy but explained it almost entirely in explicit terms of inheritance.[37]

Although polymorphism was only explicitly listed by Korson and McGregor, Booch illustrates its power on a level equal with inheritance.[38] Other sources less comprehensive than those used above also include polymorphism as an essential object-oriented element.[39][40]

This thesis acknowledges the importance of modularity but does not place it as a separate element of the object paradigm. Modularity is essential to properly

express abstract classes and to encapsulate class details which are to be hidden. As such, modularity is seen to exist in both abstraction and encapsulation by giving form to these principles.

In order to realize the benefits of the object paradigm, the object paradigm must be applied in a disciplined manner. This disciplined manner is generically termed object-oriented software development (OOSD). Methods of applying OOSD are discussed in Chapter III.

Notes - Chapter II

[1] Peter Coad and Edward Yourdon, Object-Oriented Analysis (Englewood Cliffs: Yourdon Press, 1991), 5.

[2] Bertrand Meyer, Object-Oriented Software Construction (Hertfordshire, Great Britain: Prentice Hall International Ltd., 1988), 423.

[3] R. P. Ten Dyke and J. C. Kunz, "Object-Oriented Programming", IBM Systems Journal 28, no. 3 (1989): 192.

[4] Peter Coad and Edward Yourdon, 5.

[5] John W. Verity and Evan I. Schwartz, "Software Made Simple," BusinessWeek, 30 September 1991, 92.

[6] Brad J. Cox, "There Is A Silver Bullet," Byte, Oct 1990, 209.

[7] Barry W. Boehm, Software Engineering Economics (New Jersey: Prentice-Hall, 1981), 16.

[8] Ibid., 17.

[9] Ibid., 718-723.

[10] Grady Booch, Software Engineering with Ada (Menlo Park: The Benjamin/Cummings Publishing Company, 1983), 25.

[11] Ibid.

[12] Ibid.

[13] Ibid., 26.

[14] Ibid.

[15] Ibid., 27.

[16] Coad and Yourdon, 53.

[17] Grady Booch, Object-Oriented Design With Applications (Redwood City: The Benjamin/Cummings Publishing Company, 1991), 77.

[18] Rebecca J. Wirfs-Brock and Ralph E. Johnson, "Surveying Current Research in Object-Oriented Design", Communications of the ACM, September 1990, 106.

[19] Coad and Yourdon, 1-2.

[20] Ian Sommerville, "Object-Oriented Design: A Teenage Technology," in Software Engineering for Large Software Systems, ed. B. Kitchenham (London and New York: Elsevier Science Publishers LTD, 1990), 316.

[21] Booch, Software Engineering with Ada, 38-39.

[22] Booch, Object Oriented Design With Applications, 38.

[23] Ibid., 39.

[24] Ibid.

[25] Ibid., 46.

[26] Ibid., 52.

[27] Ibid., 51.

[28] Ibid., 54.

[29] Ten Dyke and Kunz, 467.

[30] Tim Korson and John D. McGregor, "Understanding Object-Oriented: A Unifying Paradigm," Communications of the ACM, September 1990, 42.

[31] Coad and Yourdon, 30.

[32] Ibid., 53.

[33] Ten Dyke and Kunz, 467.

[34] Stefik, M. and Bobrow, D., "Object-Oriented Programming: Themes and Variations," The AI Magazine, Winter 1986, 43.

[35] Booch, Object-Oriented Design with Applications, 103.

[36] Korson and McGregor, 42 & 51.

[37] Booch, Object-Oriented Design With Applications, 54-59.

[38] Ibid., 102-114.

[39] Parker Hodges, "A Relational Successor?", Datamation, 1 November 1989, 47.

[40] Ralph E. Johnson and Brian Foote, "Designing Reusable Classes," Journal of Object-Oriented Programming, June/July 1988, 22.

CHAPTER III

OBJECT-ORIENTED SOFTWARE DEVELOPMENT

This chapter will present a disciplined approach for applying the object paradigm. The fountain life cycle is depicted as an appropriate means to implement the object paradigm. The general phases of the fountain life cycle are object-oriented analysis, object-oriented design, and object-oriented programming. This chapter also presents the business benefits of object-oriented software development.

The Fountain Life Cycle

The traditional process of developing software is through a software life cycle. A traditional software life cycle has many steps each of which has a definite starting and ending point. Each step must be completed before continuing to the next. Figure 3-1 gives Henderson-Sellers and Edwards' rendition of this life cycle.

Korson and McGregor identify problems with the traditional life cycle in that it does not allow iteration and does not emphasize reuse.[1] Booch explains iteration of processes as critical to good product development. He explains how one step in the process may

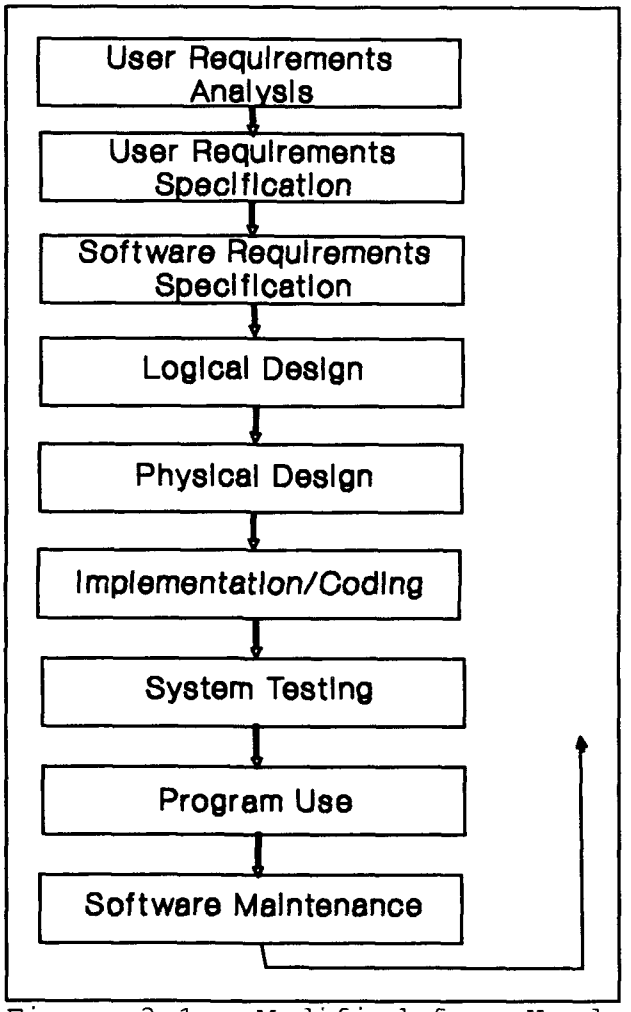


Figure 3-1. Modified from Henderson-Sellers and Edwards, The Object-Oriented Systems Life Cycle, figure 1.

identify new requirements and changes to be made. Not recognizing the need for returning to a former step "is a fundamentally poor process."[2] In Figure 3-2, Henderson-Sellers and Edwards offer their "Fountain Model" as a life cycle suitable for object-oriented software development. This life cycle recognizes the iterative aspects of an object-oriented approach and emphasizes reuse where applicable.

As Henderson-Sellers and Edwards note and Puhr and Monarchi reiterate, this model as well as the traditional approach is encompassed by three general phases of analysis, design, and implementation.[3][4] Puhr and Monarchi subdivide Figure 3-2 by associating steps one and two with analysis, steps three through five with design, and steps six through nine with implementation. Figure 3-3 represents this subdivision and incorporates the intent of Figure 3-2.

The overlapping of circles represents the blurred boundaries between the phases. Korson and McGregor explain the blurred boundaries as a result of using objects. Unlike other development techniques, objects provide a common focus in each of the development phases. The continuity of focusing on objects across analysis, design, and implementation causes a "more seamless interface between the phases."[5]

This thesis uses the compact life cycle in Figure 3-3

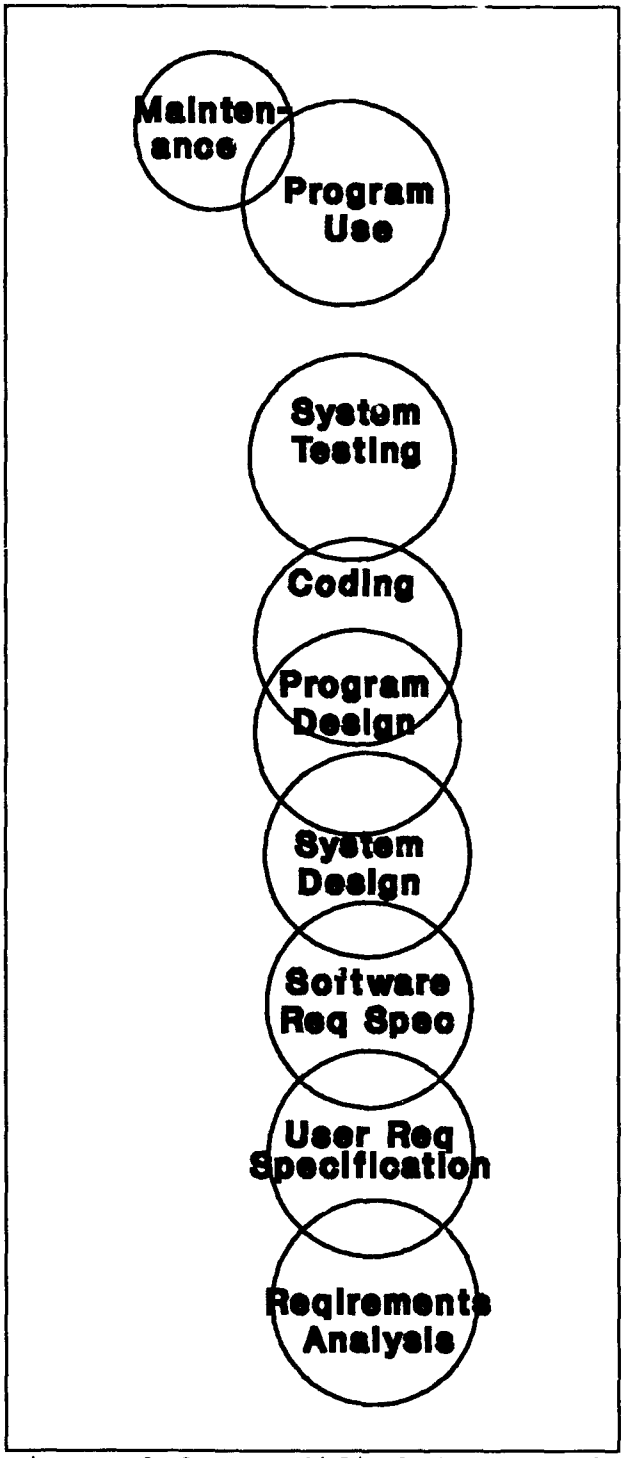


Figure 3-2. Modified from Henderson-Sellers and Edwards, The Object-Oriented Systems Life Cycle, figure 7.

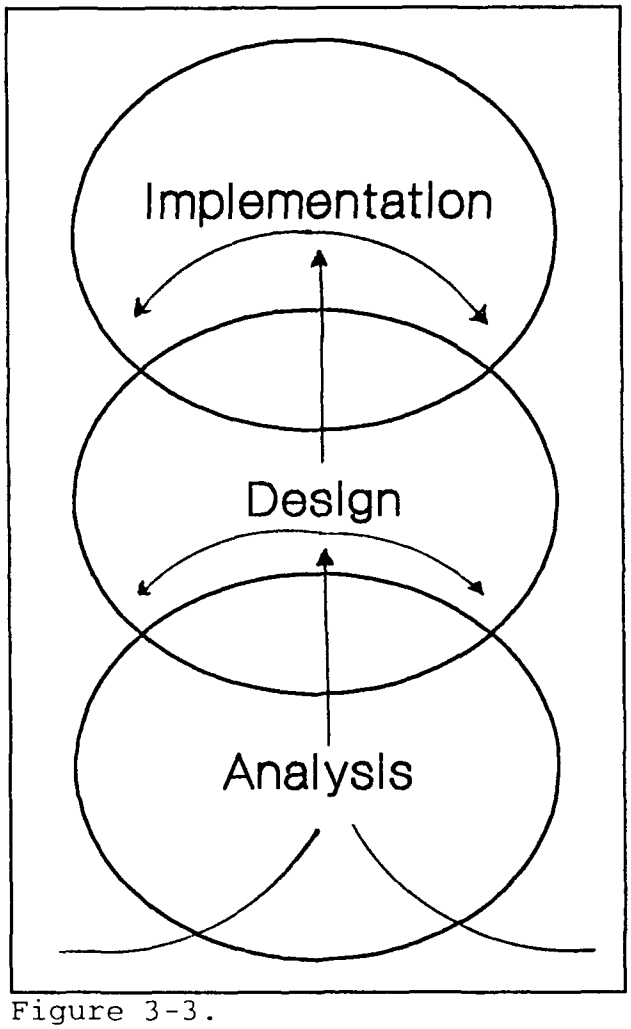


Figure 3-3.

to express the benefits of the object paradigm in each of the three broad development phases. These development phases are object-oriented analysis (OOA), object-oriented design (OOD), and object-oriented programming (OOP). This thesis will present how the object paradigm can benefit each phase via the software engineering goals.

Object-Oriented Analysis

Puhr and Monarchi point out that the purpose of systems analysis is to identify and define the problem domain and to produce an accurate representation of the problem which is to be solved with an automated system.[6] Traditional methods of analysis have not always produced suitable results. Norman identifies two problems which tend to induce less than ideal requirements analysis. First, the separate processes of data analysis and function analysis do not always converge to produce a cohesive requirements document; a point illustrated earlier by Coad and Yourdon. Second, the transition from analysis to design is not smooth due to communication problems caused by different perspectives.[7]

While not addressing specific methods, Dykman and Robbins say that much analysis documentation bears little resemblance to actual system processes. They point out many analysts fail to adequately communicate with the user in learning the business task to be automated.[8]

This thesis views this inadequate communication as a result of traditional analysis methods which do not allow easy translation from a user view of the world to an analyst view of the world. As explained below, using object-oriented analysis may reduce the impact of these problems.

Paradigm Benefits

Booch defines OOA as:

> A method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain.[9]

Obviously, users or those familiar with the real-world processes know the problem domain best. They set the vocabulary of the problem domain. Since objects have both data and operations, they allow user and analyst to view entities with a common vocabulary. The analyst can use abstraction and encapsulation to build classes and objects which can be easily understood by the user. Since all characteristics and operations can be expressed in real-world terminology, the user can more readily identify problems if any exist.

The analyst can use inheritance to establish class hierarchies so that, as Coad and Yourdon note, common attributes and services are specified once and may be extended in specific cases.[10]

Polymorphism is primarily applicable in programming since it is dependent on dynamic binding; it is planned

for in design; but, it gets its start in analysis. Anytime a user specifies an operation with the same name but in different contexts with different objects, polymorphism has begun. The analyst can identify these polymorphic functions in their contexts via the class hierarchy.

Software Engineering Support

The software engineering goals 2, 3, 4, 5, and 7 from page 11 are directly supported by OOA.

Confidence that human needs are met begins with correctly identifying those needs. Vocabulary common to both user and analyst helps accomplish this.

Precision of requirements can be achieved through precise abstract class and object specifications. The nature of these classes and objects is that a user can see exactly what will and will not be available. There is reduced confusion about which process applies to which piece of data.

When abstraction and encapsulation are applied correctly, the problem domain is expressed in a series of well-defined modules. During the iterative development process if any modifications are needed in the requirements, developers can easily make the changes to the appropriate modules.

Analyst and user time can be more efficiently used as abstract classes and inheritance allow common features to be written once. When two or more objects have common

operations and attributes, the analyst does not have to waste effort rewriting these common features. She lists only the unique features at the appropriate level.

Understandability across all development processes starts with OOA. The final analysis result is a product more easily understood by user and analyst. There is no need for confusing translation from a user view to an analyst view.

Object-Oriented Design

Puhr and Monarchi say the design stage "transforms the problem representation into a solution representation."[11] They note while the solution domain encompasses the problem domain, it may be larger. They list user interfaces, hardware control processes, and reusable domain independent objects as possible additions that must be designed into the solution.[12]

Booch identifies two key functions of OOD:

1. Object-oriented decomposition.
2. Expression of the logical and physical models.[13]

He defines object-oriented decomposition as:

> The process of breaking a system into parts, each of which represents some class or object from the problem domain . . . we view the world as a collection of objects that cooperate with one another to achieve some desired functionality.[14]

He says the logical model consists of the class and object diagrams. Effort is given to track the logical

flow of object interaction as object creation and destruction is modeled. The logical model is implementation or language independent.

The physical model evolves from the logical model and employs module and process diagrams. Module diagrams specify the actual software components to be implemented. Process diagrams address hardware components. Coad and Yourdon call this stage detailed design and say it is language dependent.[15]

Paradigm Benefits

Booch states the primary purpose of the design steps is to produce a blueprint to be used in building the solution.[16] The elements of object-orientation assist in producing this blueprint much as they do in OOA.

Korson and McGregor point out that the product of OOA is the "initial layer in the design" phase.[17] As the designers decide how to solve what is required, they will use abstraction and encapsulation to define the characteristics of user interfaces and physical resources. Internal details will be added as object attributes and functional algorithms are decided.

Henderson-Sellers and Edwards present reuse as an important part of an OOD strategy.[18] Reusable modules, developed from previous efforts, can shorten the design considerably. Designers determine the placement of these reusable modules and can extend their capabili-

ties through inheritance.

As designers study the logical flow of object interaction, they may identify needs for additional classes and objects. They may also gain a better understanding of object communication and can identify methods improved by polymorphism.

The need for additional classes usually requires clarification from users. Designers iterate back to analysis to gain additional information from the users. Objects allow users to understand the process even in the design stage due to common terminology across analysis and design.

Software Engineering Support

Object-Oriented Design directly supports the page 11 software engineering goals 1, 2, 4, 5, and 7.

The first step in providing easy to use systems is designing user interfaces which the user understands. As interface objects are designed into the solution domain, designers can take these objects to the user for review and feedback.

The identification of new classes and objects, which may have been overlooked in analysis or are required by the solution, helps ensure satisfaction of human needs. As module design increases in detail, abstraction and encapsulation can still ensure changes are localized and effects are understood.

Reuse, inheritance, and polymorphism allow design extension and deter redundant or unnecessary design details. This allows more efficient use of designer time.

As in OOA, objects allow a common dialogue to exist among user, analyst, and designer. The iterative process can be used to its fullest as the cross-flow of information is understood by each participant.

Object-Oriented Programming

Implementation follows the maturing of the design phase. This thesis applies the term object-oriented programming to cover all steps associated with implementing the design. Booch defines OOP as:

> A method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships.[19]

Programmers carry out OOP by using a suitable language to manifest the elements of the object paradigm in their code. One should note, as Duff and Howard suggest, OOP by itself does not guarantee a good system.[20] OOP must be applied by skilled programmers who understand the elements of the paradigm.

Programming can be a complex and a very error prone phase of the life cycle. Modern automated systems require enormous amounts of software which in turn requires numerous programmers. Traditionally, as these large

systems are divided into more manageable components, programmers get bogged down in communicating details with each other. They have had to overcome the problems of separating functionality and data. They have used languages and designs which force them to code their higher level modules based on code at lower levels. Beudette says these problems result in an "intertwined complexity" which is confusing and prevents reuse of previously written code.[21] As a result, he says programmers waste time rewriting the same code repeatedly.

Paradigm Benefits

The elements of the object paradigm allow programmers to handle the confusion and better manage system complexity. Objects eliminate the complexity associated with separating functionality and data. Programmers can now focus on a single component for this information.

Abstraction can play a major part in controlling complexity and aiding communication. Martin suggests that complexity is reduced in that programmers can develop their modules without having the additional burden of understanding the internal workings of another programmer's modules. They make use of other objects based only on the outward view or the interface.[22] Programmer-to-programmer communication is enhanced, also, by allowing them to discuss what their respective modules provide without forcing distracting dialogue on how capa-

bilities are provided.

Encapsulation allows programmers to hide how the abstract view is actually implemented in an object. As noted before, this allows changes to be made with little or no concern given to outside objects. Encapsulation also ensures no manipulation of an object's internal components can occur outside the messages to which an object responds.

Abstraction and encapsulation combine to provide reusability of objects. Programmers focus on the abstract view of an object (i.e., what is provided) in deciding whether to reuse it. Encapsulation provides the controlled change which may be needed to apply an object to a new solution domain. Most views of object reuse take the position that objects are reused as is, but Booch and Fischer, Henninger, and Redmiles explain this is not always the case. They point out that objects quite often need to be modified to fit a particular domain.[23][24]

Inheritance allows programmers to reduce the amount of code to keep track of. If an object needs capabilities of an established class, its class can be made a subclass of the desired class. There is then no need to rewrite the desired capabilities again.

Programming gains from polymorphism are detailed in the discussion of object-oriented elements in Chapter II.

Software Engineering Support

Object-Oriented Programming directly supports the page 11 software engineering goals 1, 2, 4, 5, 6, and 7.

Following design of proper interfaces, the programming of these interfaces allows users their first glimpse of the system in action. Users can "test-ride" the software and give their verdict on ease of use. If changes are needed, the fountain life cycle allows iteration back to either analysis or design. If the fountain life cycle and OOP are used effectively, the final product should meet the easy to use criteria.

The programmed classes and objects retain the initial vocabulary and ideas identified in analysis and design. This stable vocabulary across each development phase should allow effective communication among all participants. Good communication is critical in ensuring the proper needs are met.

Code utilizing abstraction, encapsulation, inheritance, and polymorphism is potentially far easier to modify and adapt. Time is conserved by adding modules which have to be concerned only with the outer view of any module interacted with. Programmers can make internal module changes without concern for effects on other external modules. Many capabilities can be obtained or added with little coding just by strategic placement within an inheritance hierarchy. System enhancements

using polymorphic methods can be added with little or no alterations to the message sending modules.

Not only do inheritance and polymorphism suggest easier modifications, the reduction in code required allows more efficient use of programmer time. Reusing classes and objects of previous efforts also implies reductions in the need for new code thus improving programmer efficiency.

System reliability can be increased by encapsulation and reusing modules. Encapsulating both data and operations in an object localizes any errors occurring with any enclosed algorithm. A programmer needs to focus only on the offending module for error correction. Generally, reused modules have proven reliability and including them in other systems transfers this confidence as well.

OOP continues the understanding developed in OOA and OOD. Using objects with the iterative fountain life cycle allows users, analysts, designers, and programmers to communicate with a common vocabulary which helps ensure understandability across all development processes.

Business Benefits

As explained above, the object paradigm can benefit analysts, designers, and programmers individually as well as supporting the goals of software engineering. As a

result, there are definite business benefits from which a firm may profit. This thesis finds both tangible and intangible business benefits.

Tangible Benefits

The key tangible benefit of using OOSD is reducing the time needed to develop software. Reducing development time results in reduced production costs.

Many writers point to software reuse as the major time saving feature of OOSD.[25][26][27] As stated above, reuse combined with inheritance and polymorphism can reduce designer and programmer time in developing new systems.

Time savings extend into maintenance as well. Johnson and Foote quote a National Bureau of Standards study "suggesting that 60 to 85 percent of the total cost of software" is in maintenance.[28] The controlled change allowed by encapsulation, and all other OOSD features required during maintenance, promises to greatly reduce this high cost.

For firms who sell software, the reduced time and costs allow them to market their product faster and at more competitive prices. For firms who develop software for internal use, the reduced time and costs allow them to more quickly adjust their software to changing business environments possibly gaining a substantial competitive edge.

Intangible Benefits

Firms who develop and sell software alone or as part of a larger system probably can possibly market their products and services more effectively if they employ OOSD. In today's market, identifying what the customer wants and then meeting that desire is critical to success. Applying the object paradigm through OOA defines the customer's wishes at the earliest possible time. The dialogue between the analyst and customer is in terms the customer is comfortable with. The resulting requirements documentation is in language the customer can easily identify with. The final solution/product should closely match the problem as the user sees it. This can increase customer satisfaction.

Software firms who offer an object-oriented solution versus trying to get customers to understand and sign-off on confusing functional decomposition and data flow diagrams could gain increasing market share.

Notes - Chapter III

[1] Tim Korson and John D. McGregor, "Understanding Object-Oriented: A Unifying Paradigm," Communications of the ACM, September 1990, 41.

[2] Grady Booch, Object-Oriented Design With Applications (Redwood City: The Benjamin/Cummings Publishing Company, 1991), 188-190.

[3] Brian Henderson-Sellers and Julian M. Edwards, "The Object-Oriented Systems Life Cycle," Communications of the ACM, September 1990, 145.

[4] Gretchin I. Puhr and David E. Monarchi, "Object-Oriented Analysis and Design: A Comparison of Methodologies, Techniques and Representations", Faculty Working Paper Series, College of Business and Administration, University of Colorado, Boulder, 1991, 6.

[5] Korson and McGregor, 41.

[6] Puhr and Monarchi, 7.

[7] Ronald J. Norman, "Object-Oriented Systems Analysis: A Methodology for the 1990s," Journal of Systems Management, July 1991, 32.

[8] Charlene A. Dykman and Ruth Robbins, "Organizational Success Through Effective Systems Analysis," Journal of Systems Management, July 1991, 7.

[9] Booch, Object-Oriented Design With Applications, 37.

[10] Peter Coad and Edward Yourdon, Object-Oriented Analysis (Englewood Cliffs: Yourdon Press, 1991), 15.

[11] Puhr and Monarchi, 7.

[12] Ibid., 13.

[13] Booch, Object-Oriented Design With Applications, 37.

[14] Ibid., 516.

[15] Coad and Yourdon, 180.

[16] Ibid., 37.

[17] Korson and McGregor, 41.

[18] Henderson-Sellers and Edwards, 150.

[19] Booch, Object-Oriented Design With Applications, 36.

[20] Chuck Duff and Bob Howard, "Migration Patterns," Byte, October 1990, 223.

[21] Neal E. Beudette, "Object-Oriented Programming: Untangling the Software Mess," Industry Week, 5 March 1990, 49.

[22] James Martin, "OOP Goes Beyond the Commonsense Meaning of 'Object'," PC Week, 11 September 1989, 76.

[23] Booch, Object-Oriented Design With Applications, 232.

[24] Gerhard Fischer, Scott Henninger, and David Redmiles, "Cognitive Tools for Locating and Comprehending Software Objects for Reuse," in 1991 IEEE 13th International Conference On Software Engineering by the IEEE (Los Alamitos: IEEE Computer Society Press, 1991), 323.

[25] Beudette, 52.

[26] John W. Verity and Evan I. Schwartz, "Software Made Simple," Business Week, 30 September 1991, 92.

[27] Brad J. Cox, "There Is a Silver Bullet", Byte, October 1990, 210.

[28] Ralph E. Johnson and Brian Foote, "Designing Reusable Classes," Journal of Object-Oriented Programming, June/July 1988, 24.

CHAPTER IV

THE FRAMEWORK

An effective mechanism must be used to implement OOSD in order to obtain the technology's benefits. This thesis proposes a framework of structural components as one mechanism for the effective implementation of OOSD. The framework's purpose is to present to managers a set of factors required for successful implementation of the object-oriented technology. This framework is constructed within the context of innovation diffusion theory.

The framework is presented as an idealized model. In actual practice organizations may choose to combine component responsibilities or initiate the framework in increments. The idealized model is used to present all factors in one setting.

This thesis presents the framework by identifying each component, its purpose, attributes, and operations. The support for each component and its features is derived from literature or deduced from the technology attributes.

Innovation Diffusion Theory

A common theme of most discussions of innovation diffusion theory is the S-shaped adopter distribution

curve.[1][2] The context of most S-curve dialogue is the study of firms adopting a particular innovation or technology. Brancheau and Wetherbe have adjusted the focus of the S-curve to address individuals adopting a particular technology from within a firm. In their Individual Adoption Process, they identify knowledge, persuasion, decision, and implementation as stages of thinking which people go through in deciding to "adopt or reject an innovation."[3] The knowledge stage entails awareness and understanding of the innovation. The persuasion stage involves development of a positive or negative attitude through evaluated information. The decision stage is the individual deciding to adopt or reject the innovation. The implementation stage follows a decision to adopt and is the act of committing to the innovation.[4]

The Framework Under Diffusion Theory

Brancheau and Wetherbe identify four major components of innovation diffusion theory as 1) the S-shaped adopter distribution, 2) innovativeness and adopter categories, 3) the individual adoption process, and 4) diffusion networks and opinion leaders.[5] The framework's scope focuses primarily on the individual adoption process and secondarily on the diffusion networks and opinion leaders.

Focusing on the individual adoption process, this

thesis views OOSD as too broad a technology for any one person to effectively implement in its entirety. This view necessarily requires that many individuals will be going through the stages of thinking just to reach the decision to implement OOSD or not. The framework seeks to establish the proper environment for individuals to progress through their respective knowledge, persuasion, and decision stages. The framework environment allows well-informed individuals to unite at some point in time and pool their knowledge. The firm may then act as the individual unit in the adoption process and decide to implement or reject OOSD.

Brancheau and Wetherbe define diffusion networks as the structure taken on by interpersonal communications among employees within an organization or work-group.[6] This thesis holds that a proper adoption process environment will transmit more accurate information across the diffusion networks. The result should be more positive opinion leaders should a decision be made to implement OOSD company wide.

The Framework Components

The key components necessary to establish a proper OOSD adoption process environment are: 1) The OOSD technology, 2) the Champion, 3) the OOSD User Team, 4) the Technology Resource Center, 5) Vendors, 6) the Deci-

sion Makers, 7) Corporate Goals and Objectives, 8) an Appropriate Software Project, 9) the Management Team, and 10) Communication Channels. Figure 4-1 shows a graphic representation of the components. The OOSD technology is defined in chapters II and III. The other components are described below.

The Champion

The purpose of the champion is to guide the adoption and diffusion of OOSD throughout the firm. This person is responsible for creating the proper environment and ensuring that all components interact correctly.

The champion's importance in spearheading innovation is expressed by many writers. Howell and Higgins feel "someone must take the creative idea, guide it through the trying period . . . and persevere until it becomes an innovation."[7] Beatty and Gordon studied ten companies which adopted advanced manufacturing technologies (AMT) and found that the champion was critical to the AMT projects reaching their full potential.[8] Howard and Guile explain,

> Champions are particularly important to commercialization success across a wide range of industries . . . they accept responsibility both for the focus of their enterprise and for its success and failure. They become personally invested in, and identify with, pushing whatever piece of the world they can control to . . . apply an innovative idea.[9]

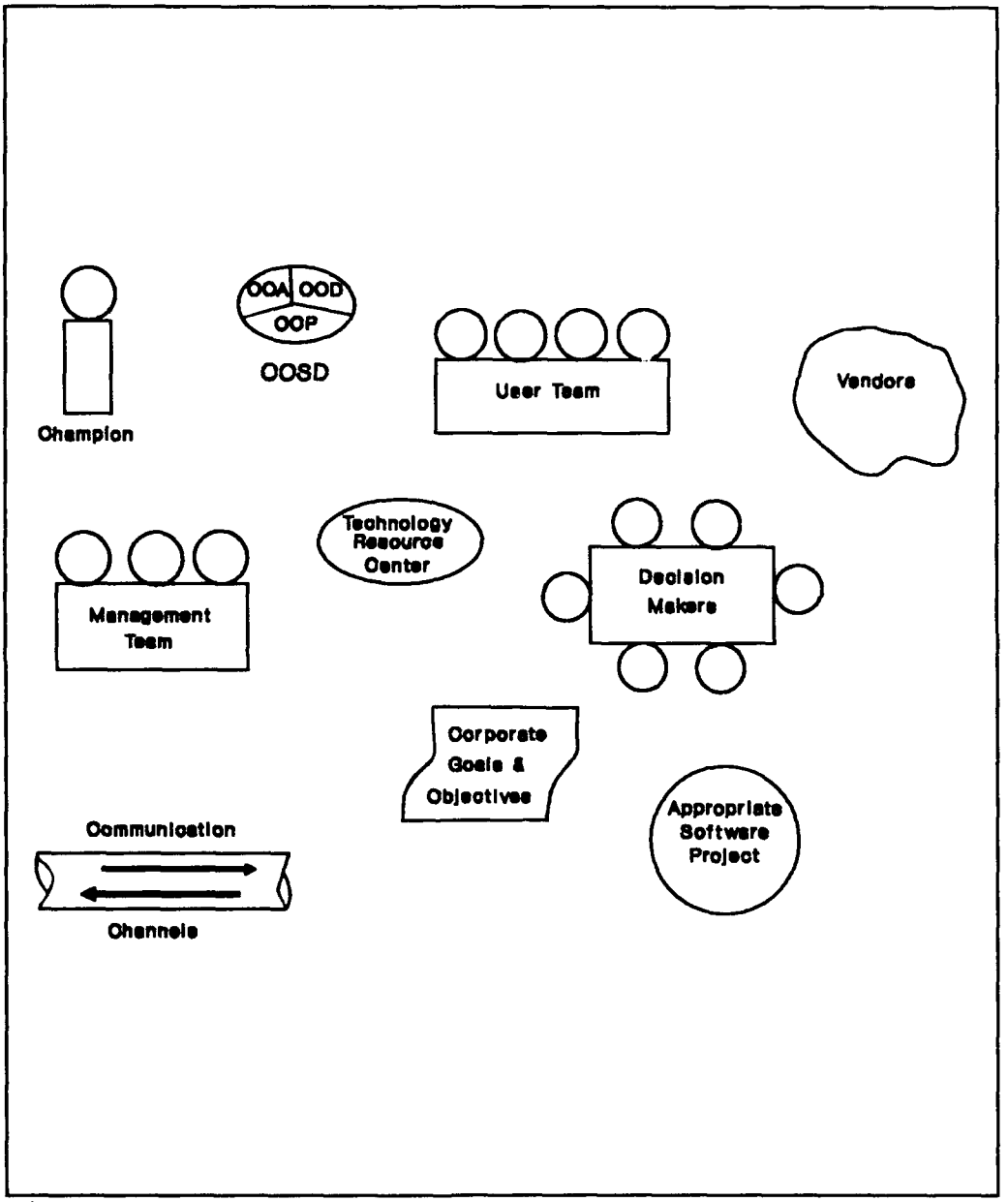


Figure 4-1. Framework Components

Table 4.1 presents this thesis's model champion to lead the implementation of OOSD.

Table 4.1

The Champion

Purpose: Guide the adoption of OOSD; bring it to life.

Attributes:

1. Positive attitude about OOSD
2. Knowledgeable in current software development methods and their limitations
3. Personality:
   a. self-confident
   b. persistent
   c. energetic
   d. risk-prone
4. High level position
5. Well connected; has open Communication Channels to many internal areas; has many external contacts in technology areas

Operations:

1. Educates self on OOSD
2. Develops understanding of how OOSD helps meet Corporate Goals and Objectives
3. Assembles pioneering User Team
4. Educates User Team in how OOSD helps meet Corporate Goals and Objectives
5. Involves User Team and Technology Resource Center in choosing Vendor OOSD methods and/or tools
6. Oversees User Team education in OOSD technology
7. Involves User Team and Technology Resource Center in choosing an Appropriate Software Project to begin building experience
8. Use Communication Channels to monitor attitudes, perceptions, and progress of User Team
9. Argues OOSD case to Decision Makers
10. Be aware of enormous short term costs associated with applying OOSD

The attributes and operations found in Table 4.1 are explained below.

Attributes

The person to lead a company into OOSD should obviously believe the technology will benefit the firm. The champion's confidence in OOSD is a key requirement in its diffusion.

In requiring the champion to be knowledgeable in current software development methods and their limitations, the model accounts for possible confrontations with those opposed to change. Properly used, the object paradigm affects all aspects of software development; changes are widespread. The champion must be able to deftly argue the merits of OOSD over those of traditional methods.

The personality attributes of self-confidence, persistence, energetic, and risk-prone were identified by Howell and Higgins. Their study found successful champions were sure of what they were doing, continued despite many obstacles, were tireless, and would readily risk their status for the benefit of the innovation.[10]

Studies by Howell and Higgins and Beatty and Gordon both identify their champions as mid-level managers.[11][12] However, this thesis takes the position that the champion should be of a higher level for two reasons. One is the large impact that implementing OOSD will have on a company. As Howard and Guile point out, the champion must be able "to garner sufficient

resources to reach the goal."[13] Therefore, the champion will need great latitude in securing the training and resources to equip the analysts, designers, programmers, and the technology resource center. Depending on the corporate structure, personnel may have to be secured from multiple departments. The required costs may require other projects go unfunded. A high level champion would have the clout needed to succeed. The second reason is the 1990's trend of eliminating mid-level positions. Companies who have trimmed their mid-level managers still need champions of innovation. Again, the leverage necessary for successfully implementing OOSD is at the higher levels not line/lower management.

The champion will need to be well-connected both within the firm and externally. The broad impact of OOSD will require the champion to influence multiple departments. Beatty and Gordon use the term evangelist for their definition of a champion.[14] The OOSD "evangelist" will definitely need people receptive to the message. Brancheau and Wetherbe say it is critical to establish effective communication channels with those who could negatively impact adoption of a technology.[15] The internally well-connected champion can use the communication channels to influence the knowledge and persuasion stages of OOSD adoption by correcting misinformation and other problems. Howell and Higgins found successful

champions were also well-connected externally.[16] Adapting their findings to OOSD, a champion would: track competitor usage of the object paradigm, seek to meet customer needs via object-oriented analysis, track trends in object-oriented tools and methods, know which suppliers best meet the firms needs, and seek information from object-oriented consultants.

Operations

This section explains the operations as identified in Table 4.1. The successful OOSD champion will exploit the above attributes by taking specific actions to reap the full benefits of the technology.

The champion must take the initiative in learning the complexities of the object paradigm. The extent of learning is determined by the corporate environment. The champion should know enough to defend and explain OOSD and its benefits to the decision makers. He/she should also have enough technical expertise to persuade those who will be developing systems with OOSD.

The champion's understanding of how OOSD helps meet corporate goals and objectives is critical if he wishes to convince others of the technology's importance to the firm's direction. Frenzel explains strategic plans supporting corporate goals and objectives are a critical success factor for information technology (IT) organizations.[17] He also notes many corporate leaders "do

not fully understand" the benefits of IT let alone specific methods.[18] The champion who can detail the importance of OOSD in strategic planning is more likely to receive the support crucial to successful implementation. Even if corporate goals and objectives do not explicitly address software development, they generally concern reducing development costs and increasing customer satisfaction. A knowledgeable champion can take the ideas presented in chapter II and III and skillfully argue OOSD's merits to corporate direction.

The champion should assemble the OOSD user team. This thesis views the user team as the means by which the firm progresses through the knowledge and persuasion stages in the adoption process. The user team is explained in the next section. Remember, in contrast to Brancheau and Wetherbe's focus on singular individuals, the broad scope of OOSD requires the firm to rely on many individuals to reach the decision stage. These individuals include analysts, designers, programmers, and others whose various areas of expertise will focus the view of OOSD throughout the firm. This thesis assumes the champion risks much in the OOSD venture. Using her clear vision of company needs, she is the one most likely to build a suitable team and maintain a proper focus.

The champion must educate potential user team members on how OOSD helps meet corporate goals and objec-

tives. Since goals and objectives guide organizational direction, team members would be more likely to accept OOSD when shown its benefits. A clear direction should also help team members seek better ways to apply the features of OOSD.

The champion should involve the user team and the technology resource center (fully defined later) in choosing vendor OOSD methods and tools. Two examples of CASE implementation indicate the wisdom of this activity. Bouldin shows that Rockwell International Corp. allowed the projected users of CASE tools to evaluate and decide which vendor product to purchase. Rockwell viewed this as a critical factor to the tool's successful implementation.[19] Norman et al. state "lack of user involvement" contributed to a failed CASE implementation at another organization.[20]

The champion should oversee the education of the user team. The outcome of the user team's efforts will have the greatest impact during the persuasion stage of the OOSD adoption process. Their success or failure in using OOSD features may irrevocably affect corporate attitudes toward the technology. Since education lays the foundation for success, the champion should ensure the user team receives adequate training. The champion should interact with the user team during training to determine if changes are needed.

The champion should again involve the user team and technology resource center (TRC) in choosing an appropriate software project for initial implementation of OOSD. This step will also increase the chance of favorable persuasion. The importance and definition of an appropriate software project are explained in a later section. Including the user team and TRC in the decision allows input to better match the available expertise to a project. A guideline here is to find a project with requirements that are well suited to the developers' previous experience. Learning the intricacies of a new application will overly complicate the task of learning OOSD.

The champion should use communication channels to actively monitor attitudes, perceptions, and progress of the user team. This operation is critical if OOSD is to have a chance to diffuse throughout the firm. Other software developers and interested parties will be watching and talking to the user team. Some in these groups will be opinion leaders. Brancheau and Wetherbe note "if opinion leaders reject a new technology, they can stop the diffusion process in its tracks."[21] It is imperative for the champion to remove barriers to progress, correct false views of the technology, and reverse negative attitudes where possible. Depending on the organization, a mix of formal and informal communication channels can be used by the champion.

The champion may have to argue the case for OOSD to the decision makers to secure their commitment. In his criticism of management practices, Tully notes:

> The capitalization, quality, staffing, and scale of the software process is not treated by corporate management as having the strategic importance which it should have, given its outputs to the organization.[22]

In their study of technology-based products, Gupta and Wilemon show a consequence of Tully's observation in that "senior management often wants innovation but they may not commit the needed resources -- time, funds, and people."[23] If the user team is to succeed with pioneering the use of OOSD, the champion must obtain the needed resources and support from senior management.

The champion must be aware of the enormous start-up costs involved with OOSD. This knowledge is crucial for planning and managing the pioneering effort. While huge costs may threaten senior management support, knowing this up front helps establish a more legitimate cost/benefit analysis. Costs involve staff training in the chosen language, programming model, and class library, conversion of existing applications, developing an organizational infrastructure for software reuse, and other organizational changes.[24][25]

The User Team

The user team's purpose is to pioneer OOSD in gener-

ating an object-oriented end product. The user team's importance is immense. It is the primary means through which the corporation as a whole will traverse the knowledge and persuasion stages and enter the decision stage in the adoption process. All eyes will focus on the team to see if the promised benefits of OOSD are real or fictitious.

Assembling a proper team is the paramount task. However, this task is challenged by many obstacles. Kellner says "software development is largely practiced as an individual creative activity, rather than a team effort."[26] He lists several forces opposed to team building: desire for autonomy, a culture that disproportionately rewards individual efforts, concentration of crucial knowledge by a few individuals, desire for privacy by developers, and large productivity differences between individuals.[27] Schlumberger feels management sees tight teams as threats to its authority.[28] An effective OOSD user team must overcome these obstacles.

This thesis presents a user team model subdivided into four components: object-oriented analysis team, object-oriented design team, object-oriented programming team, and maintenance advising team. The four components will share common attributes and operations as well as having their own specific ones. Many characteristics and

functions of the user team are similar to those of the champion. They are listed again to underline their importance to each member. Table 4.2 shows the shared user team attributes and operations.

Table 4.2

The User Team

Purpose: To generate an O-O end product.

Shared Attributes:

1. Positive attitude towards OOSD
2. Experienced software developers
3. Personality:
   a. Team oriented; works well with others
   b. Self-confident
   c. Persistent
   d. Communicates well with others
4. Professionally active outside company

Shared Operations:

1. Learn how OOSD helps meet Corporate Goals and Objectives
2. Learn principles of OOSD
3. Involved in choosing an Appropriate Software Project
4. Use Communication Channels to inform Champion of opinions, progress, and problems of OOSD project
5. Work with Technology Resource Center in building an Object Reuse Library

---

Shared Attributes

Each member assigned to the team should have a positive attitude toward OOSD. If a potential member lacks enough information to form an attitude, then at the very least he/she should posses the pioneer traits of

being venturesome and having a high tolerance for uncertainty.[29] This is, again, an ideal case. Some situations may require assigning reluctant members to the team. In these cases, the champion should observe and manage any negative impacts on teamwork.

Team members should be experienced software developers. Some in software circles disagree with this view and hold with Clarke that "it is probably easier to learn about O-O if you have no previous programming background."[30] While it may be easier to inject a new technology into an empty mind, the benefits are doubtful. This thesis stands with Gabriel and Dussud's view that while the object model is the best method for most applications it does not fit every situation.[31] In these situations, experience is essential to work a suitable solution. Also, Kinlaw says superior work teams seek to continually improve all aspects of their work.[32] Experienced software developers can determine how and why OOSD is an improvement and are better equipped to evolve into superior work teams.

OOSD user teams should have members which are team oriented, self-confident, good communicators, and persistent. While these attributes do not guarantee teamwork, members possessing them will have a greater chance of overcoming the differences which reduce productivity.

Herzog's discussion of change management shows the

importance of working well with other team members. He says during the process of change:

> The most powerful tool is . . . a sincere wish to understand the other person's point of view and to accommodate the solution that is the most appropriate for the good of the majority.[33]

Using OOSD for the first time is a major change and software developers should be team oriented and work together to find good solutions. Kinlaw suggests the amiable nature of superior work teams improves quality and efficiency.[34]

The attributes self-confident, and persistent are drawn from those of the champion. The dramatic changes of OOSD over traditional techniques make these characteristics vital to success. If problems arise with the OOSD process, the team must believe in its ability to overcome them. Kinlaw supports persistent with his description of consistency - "the long haul," as a trait of superior work teams.[35]

Members who can express themselves well are crucial if the team is to adequately discuss and overcome the problems of pioneering a new technology. As a member learns new ways to apply OOSD to the project, he/she needs to inform others and seek to cut production time.

This thesis draws "professionally active outside company" from Brancheau and Wetherbe's findings of pioneers.[36]

Shared Operations

The user team should learn how OOSD helps meet corporate goals and objectives. This is the mirror operation to the champion's task of educating the user team in goals and objectives. The champion may tell them but it is up to the user team to learn and internalize why they are using OOSD to implement the chosen software project.

The team members should learn how the principles of OOSD apply across the entire life cycle, not just to their own development area. The fountain life cycle, as described in chapter III, may at times require the entire team meet and resolve difficulties. Knowledge of OOSD's use in all development areas establishes common ground for all and should facilitate problem solving.

The user team should be involved in choosing an appropriate software project. See explanation in champion operations above.

The user team should use communication channels to keep the champion informed. Problems which cannot be handled by the team must be handled by the champion. Failure to elevate problems promptly can hinder the project and may eventually lead to wrong opinions about OOSD.

The user team should work with the technology resource center in building an object reuse library.

Biggerstaff argues that narrow domains and well understood domains/architectures are two of the key factors for successful reuse.[37] The user team is most able to explain the domain of their object modules. It can help establish storage and retrieval standards to make the most of reuse. The team is important in determining what does and does not work. A properly designed reuse library can aid future development teams and ease diffusion of OOSD.

Object-Oriented Analysis Team

The purpose of the OOA team is to "model the [system] problem domain by identifying and specifying a set of semantic objects that interact and behave according to system requirements."[38] These are the people who will talk with the users or domain experts and discover what the system requirements are. These requirements are expressed by defining appropriate objects. The OOA team has the additional attribute and operations shown in Table 4.3.

Table 4.3

The Object Oriented Analysis Team

Purpose: Model the system problem domain

Attribute:
Expertise in software requirements analysis

Operations:

Table 4.3 continued

1. Work with Champion and Technology Resource Center to choose Vendor OOA training and tools
2. Learn OOA method
3. Communicates with Users of Appropriate Software Project
4. Produces Object Oriented Requirements Specification
5. Communicates with the Object Oriented Design Team on the Object Oriented Requirements Specification
6. Revisit analysis phase as deemed necessary by the User Team

---

Attribute

Members should have some expertise in software requirements analysis. Those who have experience in systems analysis have developed needed communication skills and the ability to understand the problem domain. As they learn OOA, these members can concentrate more readily on how to apply the method without the distraction of learning other analysis skills.

OOA Operations

The OOA team should work with the champion and technology resource center to choose vendor OOA training and tools. The members' past analysis experience allows them to judge the quality of the vendor's training. They know the dimensions of the analysis task and can help detect failings of a vendor's method. Involving the OOA team can speed training and reduce the chance of additional costs to cover inadequate initial education.

Members should learn the chosen OOA method and be confident they can perform the analysis task. Any uncertainty in the beginning of OOA can only magnify future problems and jeopardize the adoption process. If a member lacks confidence in the training, this should be voiced to the champion who can then help correct the problem.

The operation, communicates with users of appropriate software project, is obvious but is included as a warning to deal directly with users/domain experts and not rely on second-hand data.

The OOA team's main function is to produce an object-oriented requirements specification. This document's importance is revealed by Coad and Yourdon. They say the product of OOA is necessary for a "progressive expansion" of the object model into object-oriented design (OOD). They further explain, "the OOA layers model the problem domain and the system's responsibilities. The OOD expansion of the OOA layers model a particular implementation."[39] In other words, OOD presents a solution to the problem as expressed by OOA and is a direct extension of the OOA model. Well formed OOA documents are critical when using the fountain life cycle as defined in chapter III. As designers and analysts revisit the OOA phase, knowing how they made past decisions will help them find new directions.

The OOA team must communicate with the OOD team on the object-oriented requirements specification. If OOD is to be the expansion of OOA as Coad and Yourdon envision it, then interaction is needed between the two teams. As shown in chapter III, the "seamless interfaces" allowed by the object paradigm gives each team common terms to aid this interaction.

To gain the benefits of the fountain life cycle, the OOA team must be available and able to revisit the analysis phase when necessary.

Object-Oriented Design Team

Drawing from Coad and Yourdon above and Puhr and Monarchi, the purpose of the *OOD* team is to produce a model of the solution domain from the objects identified in OOA. Puhr and Monarchi hold that OOD is to be language-independent.[40] However, Booch and Coad and Yourdon hold that OOD extends into physical or detail design.[41][42] Their views require addressing language dependencies toward the end of the design phase. This thesis takes the latter view. The OOD team has the additional attribute and operations shown in Table 4.4.

Table 4.4

The Object Oriented Design Team

Purpose: Produce a model of the solution domain from the objects identified in OOA.

Table 4.4 continued

Attributes:
Expertise in software design

Operations:
1. Work with Champion and Technology Resource Center to choose Vendor OOD training and tools
2. Learn OOD method
3. Use Communications Channels to talk to the OOA Team to successfully move from Analysis to Design
4. Communicate with Users of Appropriate Software Project and the OOA Team to insure proper interpretation of objects and interaction
5. Produce OOD specification
6. Revisit Design process as deemed necessary by the User Team

---

The OOD team should be experienced software designers. Although the object paradigm is vastly different from past methods, this thesis assumes experienced designers practice the goals of software engineering. As such, experienced designers should be more capable of forming an OOD which meets these goals.

OOD Operations

Many operations in Table 4.4 match those of Table 4.3. The rationale is virtually the same for these matching operations and the justification for these operations is not repeated here. As with OOA, the OOD team operations are important and should be verified by the champion or other project leadership.

The OOD team should communicate with the users of the appropriate software project when needed. While this

may not be a common practice in traditional software development, objects again allow dialogue via terms common to both parties. Booch recommends this communication saying, "often, all it takes to clear up a design problem is a brief meeting between a domain expert and a developer."[43] To ensure team unity and sustain documentation integrity, these communications should include the OOA team.

The key task of the OOD team is to produce the OOD specification. This document is used by the OOP team to code the final product. The designers should maximize the use of all object-oriented elements and emphasize reuse where possible. Doing so should allow optimum exploitation of the OOSD business benefits.

Object-Oriented Programming Team

The purpose of the OOP team is to continue the work of OOA and OOD by building the object-oriented product. The division of work is dependent upon management styles, but, normally, each programmer is assigned various classes to implement. The interaction among the resulting objects and the need to remain true to the analysis and design efforts will require the programmers to work closely together. A quality object-oriented end-product depends on the OOP team overcoming the obstacles to teamwork mentioned earlier. It is probably safe to say the unity of object interactions relies on the unity of

the OOP team. Table 4.5 shows the additional attribute and operations of the OOP team.

Table 4.5

The Object Oriented Programming Team

Purpose: Produce an object oriented product reflecting the work of the OOA and OOD phases.

Attribute:
Expertise in structured software programming

Operations:
1. Work with Champion and Technology Resource Center to choose programming language
2. Work with Champion and Technology Resource Center to choose Vendor OOP training and tools
3. Work with Maintenance Advisor to produce maintainable code
4. Learn OOP method
5. Learn programming language
6. Communicate with OOA Team and OOD Team to insure software product fits with their efforts
7. Revisit Programming process as deemed necessary by the User Team

OOP Attribute

The OOP team should be experienced in structured programming. This thesis agrees with Booch that structured programming is inadequate for large and "extremely complex systems."[44] However, structured programming is a disciplined approach and those used to such a method are more likely to adapt to the discipline of OOP. This thesis also assumes programmers familiar with structured programming are aware of its shortcomings and will

quickly accept the remedies offered by OOP. Experienced programmers will also be needed to make proper decisions in time and space efficiencies.

OOP Operations

Many of the operations for OOP are the same as those for OOA and OOD. As done previously, the Table 4.5 OOP operations comparable to OOA and OOD operations are not repeated here.

The OOP team should work with the champion and technology resource center to select the programming language. There are many topics to consider in choosing the programming language. This thesis gives substantial weight to the OOP team's opinion in the decision process based on their collective expertise. There are various languages which can emulate parts of the object paradigm. Meyer presents object possibilities and limitations with some classical languages.[45] However, as Booch notes, there presently are only a handful of truly object-oriented languages.[46] One consideration in choosing a language is deciding how many object-oriented elements the system is to utilize. If utilizing all elements is desired with the aim of reaping the most benefits, then the field of choice is narrow. Another consideration is the availability of language support tools. Flaatten et al. argue that support tool availability has a major influence on language selec-

tion.[47] The OOP team should assess tool availability for all languages and decide based upon their needs. It should be obvious that to make these considerations the OOP team will need a firm understanding of OOSD. The decision on which language to use should not be expected until well after user team OOSD training.

The OOP team should work with the maintenance advisor to produce maintainable code. McClure says, "the source code should be inspected prior to the testing phase by someone other than the original coder to audit compliance to specifications and to standards" and "to measure understandability."[48] In the context of OOSD, compliance to standards includes verifying the code is truly object-oriented and highlighting those places where code was exempted from the object paradigm. Using an object-oriented language does not guarantee object-oriented code will result. McClure explains the reason for outside inspection:

> Because of personal attachment and because of familiarity, a programmer often reads his code the way he thinks it is written rather than the way it is actually written. An impartial reader can find errors and standards violations that are not obvious to the coder.[49]

The two operations, learn OOP method and learn programming language, may be viewed by some as a single task. Training organizations tend to teach the object-oriented elements using a programming language. In instances where this happens, programmers should focus

their efforts to learn the language not as an end, but as a tool to implement an O-O product. At the conclusion of training, programmers should be confident in their knowledge and be able to apply their training in producing an adequate object-oriented product.

Maintenance Advisor(s)

The purpose of the maintenance advisor is to ensure the OOSD product can be easily maintained. Rosen defines software maintenance as "the activity that addresses itself to the correction of software errors and to remedying the inadequacies that may exist."[50] The importance of this activity is found in the fact, first cited in chapter II, that maintenance costs are 60 to 85 percent of the total software budget. Battling these costs is the maintenance advisor's first order of business. Ramamoorthy and Siyan list two maintenance functions through which the battle is fought:

> 1. *Understanding existing software.* This requires good documentation, traceability between requirements and code, and well-structured code.
>
> 2. *Modifying existing software.* This implies the need for software, . . . and data structures that are easy to expand and that minimize the side effects of changes and that have easily updated documentation.[51]

The maintenance advisor should accomplish these functions within the OOSD context of chapters II and III. Table 4.6 shows the additional attribute and operations of the maintenance advisor.

Table 4.6

The Maintenance Advisor(s)

Purpose: To insure the OOSD product can be easily maintained.

Attribute:
Expertise in software maintenance

Operations:
1. Consult with other User Team members on the maintainability of their efforts
2. Work with Technology Resource Center's Object Reuse Library on plans to retrieve and store enhanced reused modules

Attribute

The maintenance advisor should be experienced in software maintenance. Pioneering efforts, especially, need sound judgements on documentation, traceability, and the effects of change.

Operations

The maintenance advisor (MA) should consult with other user team members on the maintainability of their efforts. McClure feels "the maintainer should actively participate in the development process" throughout the project.[52] Though her argument is via traditional methods, the premise is no less fit for OOSD. The MA should be consulted throughout the fountain life cycle for an assessment on documentation and traceability of the analysis, design, and programming phases. He can

provide an outside evaluation of how well the code is structured using the object-oriented elements. He should give special attention to the use of encapsulation. Proper use of this O-O element is critical in limiting the change side-effects mentioned above.

The MA should work with the technology resource center's object reuse library on plans to retrieve and store corrected modules. Since errors may be found long after the software system is released, maintainers need to be able to easily find the flawed module and any objects useful to the correction.

The Technology Resource Center

This thesis envisions the technology resource center (TRC) as a corporate unit combining the characteristics of Prudential's Technology Transfer Center (TTC) and of a standard information center. Prudential's TTC evaluates new technologies and helps business units implement the beneficial ones.[53] The information center provides training and resources to help end-users develop their own applications.[54] Instead of end-users, the TRC targets a firm's conventional software development group. The TRC's purpose is to perform the administrative functions to obtain OOSD resources and training and to manage the formation and operation of the object reuse library. Some firms may view the TRC as equivalent to their devel-

opment center.

The TRC should be easily accessible to those needing services. Problems should be assumed possibilities as innovations such as OOSD are introduced. Giving the user team ready access to services should expedite their efforts and speed the knowledge and decision phases in the adoption process.

The TRC or a similar corporate support function is a probable entry point for OOSD. As such, the TRC will need to find a champion to promote the adoption of OOSD. The TRC should seek a person with the champion attributes listed above.

The TRC should learn the OOSD concepts. This knowledge is essential in understanding how to properly support an OOSD effort and to establish the object reuse library.

Depending on the organization, the technology resource center may have numerous departments. This thesis focuses on only two, the user team support unit and the object reuse library.

User Team Support Unit

Table 4.7 shows the pertinent attributes and operations of the user team support unit.

Table 4.7

The User Team Support Unit

Attribute:
Staff skilled in securing training resources

Operations:
1. Works with Champion and User Team in choosing a Vendor to provide OOSD training
2. Works with User Team and Champion in choosing an Appropriate Software Project

Attribute

People skilled in securing computer resources and training should staff the user team support unit. This unit should be familiar with the service quality of any potential vendor. Inadequate resources or training can negatively impact the OOSD project and impede the adoption process.

Operations

The user team support unit should work with the champion and user team in choosing vendors to provide OOSD training. The user team and champion sections reveal the rationale for this important operation. The support unit provides the administrative support for this operation.

The support unit should work with the user team and champion in choosing an appropriate software project. It

is involved in order to evaluate training requirements and resources needed.

Object Reuse Library

An object reuse library provides a collection of reusable software components. A well constructed reuse library is critical to realize the promised benefits of object reuse. Fischer, Henninger, and Redmiles explain that reuse raises "the problems of knowing when to use components, what components do, and how components should be modified."[55] A well organized reuse library should help alleviate these problems.

One should note, a pioneering OOSD effort will not have a reuse library to draw from unless a viable reuse library is purchased from a vendor. The user team may produce objects which can be used in different subsystems of the appropriate software project, but there is generally no initial library to assist in design or programming. Thus, the object reuse library will have to be built over time. This is a development project in itself and will require money and commitment from senior management.

The user team should provide input to the construction of the library. Their growing knowledge of OOSD will be invaluable in establishing the needs to be met. A careful study of Fischer, Henninger, and Redmiles' library models CODEFINDER and EXPLAINER deduces standards

which a object reuse library should meet. These standards are concerned with: meaningful classification of components, documentation of component design and original purpose, and tracking of reused and modified components.[56]

The object reuse library is included in the framework environment because of its importance in OOSD diffusion. Since reuse is a highly touted benefit of OOSD, other software developers will be observing its actual impact. Developing a reuse library early generates experience and an object inventory both of which will be useful to future projects. Resistance to the idea of reuse will be less if the benefits are observed.

Because of the high initial costs and long-term nature of the benefits associated with an object reuse library, this thesis holds that senior management commitment is critical. Banker and Kauffman's study of First Boston found it was senior management's decision to "implement a reusability approach to rebuild and upgrade the capabilities of the existing information systems."[57] Banker and Kauffman followed 20 software projects for two years. They found objects were reused across the projects an average 48 percent with a major reduction in new code needs.[58] First Boston's commitment and adequate funding allowed fulfillment of the reuse promise.

The Vendor

Within the context of this thesis, the vendor's purpose is to provide training and tools for the implementation of OOSD. The importance of selecting the proper vendors to accomplish this purpose cannot be overstated. The vendor is the avenue through which the user team will gain its working knowledge of OOSD. However, as Weber, Current, and Benton observe, literature says very little about "quantitative methods" being applied to vendor selection in any industry.[59] Their view is bolstered by the little available information on vendor selection in the University of Colorado library databases. In light of this, Table 4.8 presents an OOSD vendor based on this thesis's view of the technology's needs and the available literature support.

Table 4.8

The Vendor

Purpose: Provides OOSD training and tools

Attributes:
1. Substantiated track record
2. Skilled instructors provide training

Operations:
1. Provides proper training to appropriate personnel in OOSD methods
2. Provides proper connectivity to other vendor training and tools
3. Adapts to user's needs
4. Provides fast and effective post-purchase and post-training support when needed

Attributes

A vendor should have a substantiated track record. Many vendors' marketing literature will proclaim many companies that benefitted from their training and tools. The user team, champion, and technology resource center should be able to verify the accuracy of vendor claims. They can contact firms supported by the vendor and verify service quality. The champion's or user team's outside contacts, such as professional organizations, become valuable for verification and may become essential if project secrecy prevents contact with other firms.

The vendor should have skilled instructors to conduct training. Swander's evaluation process requires a vendor to provide instructor resumes to help judge an instructor's skill.[60] Instructors will set the OOSD foundation for the user team. To increase the probability of a quality product, OOSD trainers should have both development and training experience.

Operations

The vendor provides proper training to appropriate personnel in OOSD methods. These methods are, again, OOA, OOD, and OOP. Defining "proper training" is not an easy task. This thesis assumes adequate OOSD training equips the user team to develop a successful end product. The training should be sufficient to help overcome any unforeseen development problems. Based on these assump-

tions, this thesis views the training as proper if, after completion, the respective user team members are confident in their knowledge and can apply the training to a positive end.

The vendor should provide adequate connectivity to other vendor training and tools. If a firm chooses different vendors to train and supply tools for OOA, OOD, and OOP, problems can occur. In Kolman's analysis of CASE technology, he found companies choosing different vendor tools that were incompatible across the development life cycle. This severely limited the technology and led these firms to blame CASE itself for the problems.[61] To prevent false indictments of OOSD and to ensure smooth transitions throughout the iterative fountain life cycle, vendor training and tools must be compatible.

The vendor should adapt to user needs. Vendors should be willing to change their material and avoid an "inflexible" training regimen.[62][63] Applications such as graphical interfaces, avionics control systems, and database management systems are all different environments which may not be served well by generic tools and training.

A vendor needs to provide fast and effective post-purchase and post-training support when needed. Should the user team encounter a problem with the tools or a

question the training does not answer, fast help is essential. Poor vendor support which slows the pioneering OOSD effort may give skeptics another avenue to criticize the technology and hinder the adoption process.

The Decision Makers

The decision makers' purpose is to provide the needed resources to establish an effective environment so OOSD may be successfully integrated into the organization. The decision maker is a high level manager given broad corporate responsibilities. Depending on the organizational structure, this thesis views multiple decision makers having a say in the OOSD effort. Their characteristics, within the scope of this thesis, are presented in Table 4.9.

Table 4.9

The Decision Makers

Purpose: Provide needed resources to successfully integrate OOSD into the organization.

Attributes:
1. Is delegated authority over major resources
2. Aware of software's key role in organization

Operations:
1. Learn benefits of OOSD; recognize business opportunities
2. Find a Champion for OOSD {?}
3. Actively support the Champion
4. Communicate firm support to User Team and Technology Resource Center
5. Support the Champion, User Team, and Technology Resource Center during high level disputes

Table 4.9 continued

6. Understand OOSD's high short term costs and long term payoffs

---

Attributes

A decision maker has the authority to control her assigned resources. The extensive resources required of the framework and the changes required by OOSD demand the exercise of such authority.

The decision maker is aware of software's key role in the organization. Without this awareness the decision maker cannot be expected to appreciate the potential of OOSD and therefore be willing to devote the needed resources.

Operations

The decision makers should learn the benefits of OOSD and recognize the business opportunities. Preece lists the business and technical objectives of new technology:

1. "To increase profitability."
2. "To meet competition."
3. "To save on costs."
4. "To increase flexibility."
5. "To improve control and consistency."
6. "To achieve product improvements."[64]

A decision maker knowledgeable in OOSD's technical and business benefits could see the application to Preece's objectives and make more informed decisions on resource allocation.

Instead of the TRC, the decision maker may be the one to find the OOSD champion. Howard and Guile write of "a well-chosen . . . champion" and say "the challenge to management is to select" appropriate champions.[65] Howell and Higgins mirror this view saying, "the challenge facing management is to identify and effectively manage . . . champions."[66] If OOSD knowledge first enters through a decision maker and gains tentative acceptance, then the decision maker should be involved in finding a champion to lead the OOSD effort.

There should be at least one decision maker actively supporting the champion. The champion will have little success pioneering OOSD if adequate resources are not available. Lack of senior management support says that the organization has serious doubts in the technology. Opinion leaders outside the OOSD project are more likely to favor the technology if decision maker support is firm and visible.

The decision makers should communicate support to the user team and technology resource center also. This thesis views the user team and TRC as sharing risk in pioneering an OOSD effort. Decision maker support is

needed to overcome the anxiety associated with this risk. Support communicated to this level can positively affect diffusion networks as well. Howell and Higgins hold that visible senior level support "can help . . . overcome active opposition, passive resistance, or indifference to the innovation."[67]

The decision makers should support the champion, user team, and technology resource center during high level disputes. Howell and Higgins note, "many projects have been shelved because of political in-fighting" and other disputes but executive protection "from these negative influences . . . can facilitate the innovation process."[68] They argue protection for the champion only but this thesis expands the coverage to the user team and TRC due to the level of risk they enjoy.

The decision makers should understand the nature of OOSD's high short term costs and long term payoffs. As discussed for champions and reuse libraries, initial costs of OOSD are high. The payoffs come over time as workable reuse libraries are built, maintenance costs are cut, and the other previously noted benefits are realized. This knowledge allows management to make informed investment decisions and helps alleviate unrealistic expectations.

Corporate Goals and Objectives

The purpose of the corporate goals and objectives is

to generally provide direction for the firm and to specifically address software concerns. Information on goals and objectives is abundant. This thesis highlights only the features essential to diffusion of OOSD. Table 4.10 presents the corporate goals objectives structure.

Table 4.10

The Corporate Goals and Objectives

Purpose: Provide direction for corporation

Operations:
1. Address computer resources
2. Require the benefits which OOSD can provide
3. Provide unifying guidance for all corporate activities

Attribute:
Specific and easily understood

Operations

The goals and objectives should address computer resources. Firms which specifically address computer resources in their goals and objectives acknowledge the importance of these assets and the people who work with them. This thesis assumes the companies who market hardware and software do this but those firms in which computer resources are a supporting function may not address this asset's significance. This operation is a critical first step toward a meaningful evaluation of OOSD.

The goals and objectives should require the benefits

which OOSD can provide. Giegold says key functions of an objective are "to record the commitment of . . . the organization to achievement of a needed improvement" and "to motivate the performer."[69] If the goals and objectives do not specify a need which OOSD can fill, then there is no reason to invest in the technology. People will be motivated to consider and apply OOSD if they can see its application to a required improvement.

The goals and objectives should provide unifying guidance for all corporate activities. Pioneering an OOSD effort would be the job of Frenzel's information technology (IT) organization. He argues defining the purpose and roles of the IT organization within the company is critical to the IT organization's success.[70] This operation would show the champion, user team, technology resource center and others their purpose and contribution to the company as a whole.

Attribute

A goal or objective must be specific and easily understood says Giegold.[71] This attribute is stated to emphasize that goals and objectives should be meaningful and known through all levels. Without this, the above operations will lack clear direction.

The Appropriate Software Project

The purpose of the appropriate software project is

to provide a vehicle for building expertise in OOSD. Implementations of other technologies provide guidance on choosing an appropriate software project. In discussing the implementation of an information architecture, Brancheau and Wetherbe advise:

> Build an island of success. Take on a small but visible pilot project. . . . Find the right project . . . that will give short-term benefits and . . . use the project to build experience and gain credibility.[72]

To introduce CASE, Gray, Brancheau, and Kozar say the project should be small but "of sufficient size to test the limits of the CASE product."[73] The Goddard Space Flight Center Software Engineering Laboratory used a pilot project named GRODY to experiment with OOSD. This project "provided an extremely valuable experience in the application of object-oriented principles to a real system."[74] They used the experience to create their own OOSD method GOOD, which "is now being used on an increasing number of projects inside and outside of the Goddard Space Flight Center."[75]

Modeled after the above information, Table 4.11 presents the structure of the appropriate software project.

Table 4.11

The Appropriate Software Project

Purpose: Vehicle to begin building expertise in OOSD

Attributes:
1. Non-critical to immediate corporate success
2. Similar in size and purpose to previous projects
3. Choice is agreed upon by Champion, User Team, and Technology Resource Center

OPERATIONS:
1. Provides ample opportunity for practicing learned methods of OOA, OOD, and OOP
2. Lays the foundation for the Reuse Library
3. Provides Champion opportunity to study and adjust management methods in using OOSD

Attributes

The appropriate software project is non-critical to immediate corporate success. There is already a great deal of pressure in learning and applying OOSD principles. The added stress of keeping the company out of the red may tempt the user team to take short-cuts and forego OOSD and return to traditional methods.

An appropriate software project is similar in size and purpose to previous projects worked on by the user team members. This attribute should allow the project to "test the limits" of OOSD and be visible enough to gain credibility among management and opinion leaders. This should also relieve some user team anxiety in that the actual product is not something totally foreign.

The champion, user team, and TRC agree on the choice

of an appropriate project. The rationale for this attribute is found in the respective sections above and is not repeated here.

Operations

An appropriate software project provides ample opportunity for the user team to practice OOA, OOD, and OOP. As in the Goddard Space Flight Center example, this operation lets the user team work out the kinks of the new technology and gives them needed expertise to be applied to future, more critical projects.

The project lays the foundation for constructing the reuse library. As the user team begins to master OOSD they will create an inventory of objects. This gives the TRC the opportunity to research the problems and solutions of reuse and to begin building a workable library.

The project provides the champion the opportunity to study and adjust management methods in using OOSD. Just as OOSD requires software developers learn new technical practices, new management practices are required also. The fountain life cycle will require managers to be far more flexible than the rigid techniques demanded of the waterfall life cycle. A reuse library will open many opportunities to cut development costs but will tax management with its conflicting control and easy access requirements. The champion can use the project to compile management information to be used on future critical

products.

The Management Team

The purpose of the management team is to assist the champion in implementing OOSD. The management team is a component whose existence depends on the program management style of the organization. If the champion does not have direct authority over the user team and persons assigned from the TRC, then the champion must work through other managers. These other managers form the management team.

The attributes and operations of the management team are a subset of those of the champion. Except for being high level and well-connected, the management team should possess the champion attributes. Any negativism by the managers could adversely affect the user team and jeopardize the adoption process. The champion may delegate or seek assistance in the performance of the champion operations.

The Communication Channels

The purpose of the communication channels is to allow effective transfer of all needed information concerning organizational use of OOSD. This component lacks the physical constraints of the other components and is not as easily observed. As such, the communication channels are not presented with attributes and opera-

tions. Though communication channels are not as visible, they are vital and can be tested when properly understood.

Two of Drucker's communications principles are "communication is perception" and "communication makes demands."[76] He explains communication is perception as the communication process being controlled by the receiver of information. It is the receiver's perception of the information which decides if the communication is successful. The receiver must perceive the information as it is intended by the communicator. Success is not assured just because the receiver heard the communicator. He explains the principle "communication makes demands" by saying communication,

> Always demands that the recipient . . . do something, believe something. . . . If it goes against his aspirations, his values, his motivations, it is likely not to be received . . . or to be resisted.[77]

Many of the framework operations require communications among company members. These communications will occur through either formal or informal communication channels. This thesis views these communication channels as vital in getting correct information to the right receivers. In following Drucker's principles, these recipients must interpret the information correctly. Both communicator and receiver should understand the "demands" made by the communication.

As communication occurs in an OOSD development

environment, the champion or management team can monitor the information flow and judge the resultant opinions and actions taken. The champion or management team can then make corrections if problems occur due to incorrect perceptions or improper development actions.

Summary

This framework is composed of ten components consisting of various attributes and operations which mark the implementation factors required by OOSD. Current literature supports many of the constructs proposed in this chapter. Organizations seriously considering implementing OOSD may greatly benefit by adapting this framework to their existing structure. This thesis strongly contends that application of such a framework greatly enhances the knowledge and persuasion stages and leads to a more informed decision stage during the OOSD adoption process.

Notes - Chapter IV

[1] James C. Brancheau and James C. Wetherbe, "Understanding Innovation Diffusion Helps Boost Acceptance Rates of New Technology", Chief Information Officer Journal, Fall 1989, 23.

[2] John Hagedoorn, The Dynamic Analysis of Innovation and Diffusion (London: Pinters Publishers Limited, 1989), 120.

[3] Brancheau and Wetherbe, 26.

[4] Ibid., 26-27.

[5] Ibid., 23.

[6] Ibid., 28.

[7] Jane M. Howell and Christopher A. Higgins, "Champions of Change: Identifying, Understanding, and Supporting Champions of Technological Innovations," Organizational Dynamics, Summer 1990, 40.

[8] Carol A. Beatty and John R. M. Gordon, "Preaching the Gospel: The Evangelists of New Technology", California Management Review, Spring 1991, 74.

[9] William G. Howard, Jr. and Bruce R. Guile, Profiting from Innovation: The Report of the Three-Year Study from the National Academy of Engineering (New York: The Free Press, 1992), 92.

[10] Howell and Higgins, 41.

[11] Ibid., 44.

[12] Beatty and Gordon, 73.

[13] Howard and Guile, 93.

[14] Beatty and Gordon, 80.

[15] Brancheau and Wetherbe, 28-29.

[16] Howell and Higgins, 47.

[17] Carroll W. Frenzel, Management of Information Technology (Boston: Boyd & Fraser Publishing Company, 1992), 87.

[18] Ibid., 89.

[19] Barbara M. Bouldin, "Automate your software development with minimum pain," EDN, 7 June 1990, 111.

[20] Ronald J. Norman, Gail F. Corbitt, Mark C. Butler, Donna D. McElroy, "CASE Technology Transfer: A Case Study of Unsuccessful Change," Journal of Systems Management, May 1989, 37.

[21] Brancheau and Wetherbe, 29.

[22] Colin Tully, "A Failure of Management Nerve and Vision," in 1991 IEEE 13th International Conference On Software Engineering by the IEEE (Los Alamitos: IEEE Computer Society Press, 1991), 155.

[23] Ashok K. Gupta and David L. Wilemon, "Accelerating the Development of Technology-Base New Products," California Management Review, Winter 1990, 34.

[24] Chuck Duff and Bob Howard, "Migration Patterns: Moving to object-oriented technology is more involved than simply buying a compiler," Byte, October 1990, 224.

[25] Anthony I. Wasserman, "Object Insider: Object-Oriented Thinking," Object Magazine, September/October 1991, 10.

[26] Marc I. Kellner, "Non-Technological Issues in Software Engineering: Panel Session Overview," in 1991 IEEE 13th International Conference On Software Engineering by the IEEE (Los Alamitos: IEEE Computer Society Press, 1991), 146.

[27] Ibid.

[28] Maurice Schlumberger, "Software Engineering Management," in 1991 IEEE 13th International Conference On Software Engineering by the IEEE (Los Alamitos: IEEE Computer Society Press, 1991), 153.

[29] Brancheau and Wetherbe, Figure 3, 26.

[30] Dan Clarke, "Object Insider: Diskette frisbee isn't fun," Object Magazine, May/June 1992, 19.

[31] Richard P. Gabriel and Patrick Dussud, "Neoclassical object-oriented techniques: multimethods," Object Magazine, May/June 1992, 41.

[32] Dennis C. Kinlaw, Developing Superior Work Teams (Lexington: Lexington Books, 1991), 16 & 59.

[33] John P. Herzog, "People: The Critical Factor In Managing Change," Journal of Systems Management, March 1991, 11.

[34] Kinlaw, 14.

[35] Ibid., 15-16.

[36] Brancheau and Wetherbe, 26.

[37] Ted J. Biggerstaff, "Software Reuse: Is It Delivering?," in 1991 IEEE 13th International Conference On Software Engineering by the IEEE (Los Alamitos: IEEE Computer Society Press, 1991), 53.

[38] Gretchin I. Puhr and David E. Monarchi, "Object-Oriented Analysis and Design: A Comparison of Methodologies, Techniques and Representations", Faculty Working Paper Series, College of Business and Administration, University of Colorado, Boulder, 1991, 13.

[39] Peter Coad and Edward Yourdon, Object-Oriented Analysis (Englewood Cliffs, N.J.: Yourdon Press, 1991), 178.

[40] Puhr and Monarchi, 13.

[41] Grady Booch, Object-Oriented Design With Application (Redwood City: The Benjamin/Cummings Publishing Company, Inc., 1991), 157.

[42] Coad and Yourdon, 181.

[43] Booch, 143.

[44] Ibid., 18-19.

[45] Bertrand Meyer, Object-Oriented Software Construction (Hertfordshire: Prentice Hall International (UK) Ltd, 1988), 375-382.

[46] Booch, 36.

[47] Per O. Flaatten, Donald J. McCubbrey, P. Declan O'Riordan, and Keith Burgess, Foundations of Business Systems (Orlando: Dryden Press, 1989), 158.

[48] Carma L. McClure, Managing Software Development and Maintenance (New York: Van Nostrand Reinhold Company, 1981), 56.

[49] Ibid.

[50] Anthony Ralston and Edwin D. Reilly, Jr., eds. Encyclopedia of Computer Science and Engineering (New York: Van Nostrand Reinhold Company Inc., 1983), s.v. "Software Maintenance," by Saul Rosen.

[51] Anthony Ralston and Edwin D. Reilly, Jr., eds. Encyclopedia of Computer Science and Engineering (New York: Van Nostrand Reinhold Company Inc., 1983), s.v. "Software Engineering," by C. V. Ramamoorthy and K. Siyan.

[52] McClure, 40.

[53] Nicole A. Wishart and Lynda M. Applegate, "The Prudential: Organizing For Technology Innovation," Harvard Business School Case Study (Boston : Harvard Business School, 1990) 4.

[54] Flaatten, McCubbrey, O'Riordan, and Burgess. 64-65.

[55] Gerhard Fischer, Scott Henninger, and David Redmiles, "Cognitive Tools for Locating and Comprehending Software Objects for Reuse", in 1991 IEEE 13th International Conference On Software Engineering, by the IEEE (California: IEEE Computer Society Press, 1991), 323.

[56] Ibid., 319-326.

[57] Rajiv D. Banker and Robert J. Kauffman, " Reuse and Productivity in Integrated Computer-Aided Software Engineering: An Empirical Study," MIS Quarterly, September 1991, 378.

[58] Ibid., 388-390.

[59] Charles A. Weber, John R. Current, and W.C. Benton, "Vendor selection criteria and methods," European Journal of Operational Research 50, no. 1 (January 1991): 16.

[60] Alice J. Swander, "Vendor Training: Establishing the Evaluation Process," Performance & Training 29, no. 4 (April 1990): 33.

[61] Joe Kolman, "Getting down to CASEs," Institutional Investor, August 1990, 120.

[62] Swander, 33.

[63] Charles A. Litchfield, "Vendor Training: A Question of Commitment to User Success," Journal of Library Administration 12, no. 2 (1990): 9.

[64] David A. Preece, Managing The Adoption Of New Technology (London: Routledge, 1989), 15-31.

[65] Howard and Guile, 92-93.

[66] Howell and Higgins, 55.

[67] Ibid., 53.

[68] Ibid.

[69] William C. Giegold, Management By Objectives: A Self-Instructional Approach, vol. 1, Strategic Planning and the MBO Process (New York: McGraw-Hill, 1978), 15.

[70] Frenzel, 79.

[71] Giegold, 15.

[72] James C. Brancheau and James C. Wetherbe, "Information Architectures: Methods and Practice," Information Processing and Management 22, no. 6 (1986): 463.

[73] Linda Gray, James C. Brancheau, and Kenneth A. Kozar, "Evaluation, Environment, and Education: The 3E Framework for CASE Implementation," Software Engineering: Tools, Techniques, Practices, March/April 1992, 18.

[74] Ed Seidewitz, "General Object-Oriented Software Development: Background and Experience," The Journal of Systems and Software 9 (1989), 107.

[75] Ibid.

[76] Peter F. Drucker, Management: Tasks, Responsibilities, Practices (New York: Harper and Row, 1974), 483.

[77] Ibid., 487.

CHAPTER V

RESEARCH APPROACH

This chapter presents how the research was conducted. It covers the chosen research method, the process of contacting companies, how participants were selected, rationale of the question set design, and the goal of the research process.

Research Method

Nachmias and Nachmias discuss three methods of data collection: the mail questionnaire, the personal interview, and the telephone interview.[1] I chose the personal interview as the primary data collection tool. I used the telephone interview to gather limited information to judge whether or not a company was a prime candidate for study.

The disadvantages of the mail questionnaire are it "requires simple questions," provides "no opportunity for probe," gives "no control over who fills out the questionnaire," and has a "low response rate."[2] To fully test the pertinence of the framework components, I needed more than simple questions. I also needed the ability to ask more probing questions based on any given response. Since I was originally attempting to do a detailed study of only two companies, I could not afford

a low response rate.

The telephone interview would have severely limited answers in the environment of the one company that committed to my research. Nine of the eleven people interviewed worked in cubicles or shared offices. Others around them could easily hear them talking on the telephone.

Of the three types of personal interviews presented by Nachmias and Nachmias, I chose the nonscheduled-structured interview. Its four characteristics meshed well with the nature of my research:

1. It takes place with respondents known to have been involved in a particular experience.
2. It refers to situations that have been analyzed prior to the interview.
3. It proceeds on the basis of an interview guide specifying topics related to the research hypothesis.
4. It is focused on the subjective experiences regarding the situations under study.[3]

The common experience shared by the participants was developing a software system using OOSD. I was able to do limited analysis before the interviews to ensure their situations fit within the scope of my research. The subjective experiences were important in that respondents' subjective interpretations of the activities involved with implementing OOSD have great influence on its possible diffusion.

Contacting Companies

I first contacted ten companies who had participated in a past mail survey conducted by the Emerging Technologies Project, University of Colorado. These ten companies indicated on the survey they were actively involved in object-oriented development or research. However, none of these firms wanted to participate in my on-site case study.

I then compiled a list of suitable companies based on available literature and sought help from several professional contacts of mine. I, or persons on my behalf, contacted 29 firms over a period of three months. The locations ranged from Massachusetts to California and Michigan to Texas and many points in between. Of these, eight agreed to consider my proposal and review the project details description in Appendix C. Only one company committed to participate in the study.

Participant Selection

I asked my contact at the firm to request volunteers to be interviewed. Of thirteen people I hoped would volunteer, eleven did so. The positions of the eleven are two analysts, one designer, three programmers, one mid-level manager, one support person from their equivalent of the technology resource center, one senior manager, and two "champions."

Although neither of the two people identified as champions met the full definition as shown in chapter IV, they did perform some champion operations. I decided to interview them with the champion question set and see what the results would be. One champion was not associated with the studied project and was interviewed with a modified champion question set.

Question Set Design

The question sets in Appendix A were designed to test for the presence of the attributes and the performance of the operations of the framework components and to test for realization of the OOSD benefits. Often the same question, or a rephrasing, was designed into different question sets. This allowed testing for consensus or conflicting views on such items as communication channels, OOSD technology, training, etc.

Process Goal

The original goal of the research was to study the implementation of OOSD at two firms and compare their experiences with this thesis's framework. Two detailed case studies would begin the process of compiling data which could be used to adjust the framework or suggest its soundness. Detailed studies of several other companies are required to secure confidence in the framework.

Time limits and the need to study OOSD in depth restricted the goal sample to two firms. Company reluctance to participate limited the actual sample to one firm. However, though only one company participated, valuable information was discovered in their implementation of this emerging technology.

Notes - Chapter V

[1] David Nachmias and Chava Nachmias, Research Methods in the Social Sciences (New York: St. Martin's Press, 1981), 181.

[2] Ibid., 182-183.

[3] Ibid., 191.

CHAPTER VI

ANALYSIS and FINDINGS

This chapter analyzes the case study data in reference to the proposed OOSD framework. Reviewing the tables in chapter IV will greatly aid the reader's comprehension of this analysis. Presented is background information on the studied OOSD project followed by examinations within the context of each framework component. A brief summary of the key findings concludes this chapter.

The analysis will refer extensively to the interview data in appendix B. Rather than repeat the information fully, the data is referenced by person and answer number. Some answers were very voluminous and required separation by paragraphs. A paragraph number will be used in such cases. Example: [Senior Manager: 4a-para. 3].

Case Study Project Background

This section presents a brief sketch on the company structure and the project history. Due to the requirement that the company remain anonymous, very little specific information will be revealed. What is revealed

is enough information to give the reader the context in which the OOSD effort took place.

Company Structure

The company studied is one of many firms engaged in developing software for business applications. Depending on one's viewpoint, this company has a tall management hierarchy. The case study location had four levels of management above the staff level development team.

This company, like many other firms, has placed its software analysts within its marketing hierarchy. The analysts deal directly with the customer or domain experts. The software designers and programmers are part of the product development hierarchy. Though the analysts are considered part of the development team, they are not under the reporting authority of product development.

Software development is the primary function of the case study location. The interview data reveal estimates of 70 to 90 percent of all work is in developing software. The company has other locations which research new software development methods including OOSD. The case study site has found little application for this research. Methods of choice for marketable product development at this location are functional decomposition techniques.

Project History

As stated in Appendix B, AFCAP is this thesis's masked acronym for the studied project.

AFCAP is a major version upgrade of a previous software release. The previous version was developed using traditional functional decomposition. AFCAP is this site's pioneering effort at applying OOSD to a revenue generating product.

The AFCAP project has approximately 450,000 lines of code. The original manning was 10 people which increased to a maximum of 20 at one point in the product's development. C++ was the object-oriented language used in the applications. The original scheduled time-to-develop was approximately 1.5 years. Actual completion time was three years.

Initial contacts with the company suggested the AFCAP project was a success story of OOSD usage; however, as the data suggest, this is not necessarily clear.

If one defines success as delivering a product which satisfies the customer, then AFCAP is a success. The development team completed the project and is testing the system at several customer sites as of this thesis writing. Initial customer reaction to the product is very favorable.

If one defines successful OOSD use as delivering a near total object-oriented product, then AFCAP is not a

success. The interview data indicate some parts of AFCAP were developed with traditional techniques. User team inexperience with OOSD combined with time problems resulted in a departure from OOSD in some instances. The project was late and will not generate the expected profits.

The following analysis presents a more detailed picture of AFCAP's development.

OOSD Technology

This section reveals the extent of usage of OOSD by the firm. The analysis is divided into OOSD principles, life cycle approach, and benefits realized.

OOSD Principles

Judging the extent of use of the OOSD principles in OOD separate from OOP is virtually impossible with this case study. The method of design and programming employed by the user team was to have the same person do the design and programming of a software unit.[Designer: 19 & 20][Programmer #1: 19c][Champion #1: 8c] Since this was the case, this paper will assume the extent of use in programming was also reflected in the design.

Each programmer used objects extensively. Two programmers used abstraction and encapsulation extensively with one reporting only moderate use. Only one programmer used inheritance extensively with the other two

reporting moderate use. All three programmers used polymorphism very little.[Programmer #1, 2, 3: 16a-e] One programmer even expressed a dislike for polymorphism.[Programmer #1: 16e]

The less than total use of object-oriented principles led to a design and end product that was a mixture of traditional and object-oriented methods. Of those highlighting this fact, two cited time constraints and one inadequate training as the reasons for the problem.[Champion #1: Inserted Question II][Programmer 2: 2a][Manager: 6c]

Object-oriented analysis was not applied to this project. None of the OOA principles were used in defining the problem domain. The analysts were unfamiliar with OOSD and developed non-object-oriented requirements to be used by the designer-programmers.[Analyst #1: 1 & 2b][Analyst #2: 1]

Life Cycle Approach

While the iterative or fountain life cycle is the method of choice for OOSD, this effort's use of it was an unplanned occurrence. Each of those citing an iterative life cycle said its use was out of necessity due to continually changing requirements.[Analyst #1 and #2: 13a][Designer: 13] Analyst #1 and Designer said OOSD did not *cause* the use of an iterative life cycle; Analyst #2 was unsure. Designer stated her displeasure with using

an iterative approach. However, Analyst #1 felt the traditional waterfall life cycle "doesn't work in complex systems."

Benefits

As stated previously, combining the object-oriented principles, a fountain life cycle, and reuse practices promises to decrease development time and costs and to improve relations with users. Logic implies a pioneering or early OOSD effort will not fully realize these benefits due to large start-up costs. The case study seems to support this.

Two people could see no real benefits of OOSD.[Senior Manager: 4a, para. 1][Analyst #1: 2a] Of those finding a benefit in this project, all cited improved maintainability. Errors were more easily corrected and enhancements were inserted without propagating errors through other code.[Champion #1: 6a, 6b][Designer: 2c] [Programmer #1, #2, #3: 2b] Champion #1 felt the quality had improved. Programmer #3 felt reusability and inheritance also helped him do his job. One should note this form of reusability is only reusing modules from within the project.[Programmer #2: 12b] This project did not realize the full benefit of reuse. The provided library was of limited value and contributed little to the effort.[Designer: 12a][Programmer #2, #3: 12b][Manager: 6e][User Support: 16]

The Champion

The case study found no person matching the champion framework component during the research. However, two people were identified as staunch advocates of OOSD. One directly influenced OOSD's use in the AFCAP project and the other semi-regularly pushed the technology in other areas of the firm. Neither enjoyed the clout required by the framework in that both are staff-level employees and senior management apparently could not identify them by name.[Champion #1, #2: 3][Senior Manager: 6] Their limited influence is revealed in the following analysis. Due to Champion #2 having no relationship with the AFCAP project, the analysis is split into two parts.

Champion #1

This champion had a positive attitude toward OOSD.[#1: 1] He seemed to understand the advantages of the object paradigm over traditional methodologies [#1: 1] but did not understand the full application of the technology. He did not have a grasp of the importance of OOA.[#1: 6d]

Since this person did not meet the literature definition of a champion, I decided not to seek personality data.

This person cannot be considered well-connected. Communication channels are not very open to him and he

does not deal with people outside his work area. Time constraints at his job level are stated as the limiting factor. He is associated with only one professional organization. He does not receive technical information from the few professional contacts he has.[#1: 14c, 17, 18, 19a&b]

Of the ten champion component operations, this person performed four to some degree. He educated himself on the technology and experimented with its use.[#1: 5b] He sees how OOSD helps meet his company's goals and objectives.[#1: 6a] Although he did not argue OOSD's case to the decision makers, he was obviously instrumental in securing its use in the project against opposition: others saw him as a champion in the beginning; he and one other were the only ones totally for OOSD; other projects tried to stop the use of OOSD.[#1: 2, 9, 7c] He, along with management, became aware of the large start-up costs but none seemed aware at the beginning of the project. This is evidenced by the severe underestimates in needed training time and resources.[#1: Inserted Question II, 15c]

The other six operations were responsibilities not given him or were not performed at all. The project leader was his superior.[#1: 16a, 8b, 12b, 10a&b, 11, 15a&b, 16b]

Champion #2

Champion #2 was not associated with the AFCAP project and was interviewed with the modified champion question set.

Champion #2 believes in OOSD but is not very optimistic about its immediate future in the company.[Champion #2: 1a, 1b]

As with Champion #1, Champion #2 is also a low level employee.[Champion #2: 3] He disagrees with the framework's view that a champion should be at a higher level position. He feels pushing OOSD at the lower levels will bring a better chance of success.[Champion #2: 4a] The data suggests, however, that his way is not working.

Champion #2 is better connected than Champion #1, but he is not receiving many benefits from his connections.[Champion #2: 8, 9]

Although he has not influenced the user team, he has presented the OOSD message to others in the firm. He reports lecturing on OOSD to all levels of the company. He continues to find divided opinions about the technology.[Champion #2: 2a] He knows of no one at senior levels who are OOSD advocates.[Champion #2: 4b]

User Team

The firm used a team approach to develop the AFCAP system, but as discussed, the entire team did not apply

OOSD. Although the analysts were team members, they did not have the same chain of command as the other developers.[Analyst #1, #2: 10]

This section evaluates the presence or absence of the user team attributes and operations. This information is presented via collective features, OOA team, OOD team, OOP team, and maintenance advisor.

Collective Features

At the start of the project only two of the ten original team members had a positive attitude toward OOSD. Two were opposed and eight were ambivalent.[Champion #1: 9] Champion #1 and Manager both feel the entire team has come to prefer the technology.[Champion #1: 9][Manager: 7a] The data, however, reflect a mixture of attitudes.[Analyst #1: 1][Designer: 2a][Programmer #1: 2a&b][Programmer #2: 2b][Programmer #3: 2b, 7] Analyst #1 has developed a negative attitude based on insufficient information. Designer and Programmers #1 and #2 have a positive attitude. Programmer #3 questions its usefulness to some problems but does acknowledge positive benefits.

All team members sampled had prior experience with software development projects. The least experienced member worked only three projects including this one. [Analyst #1, #2: 3a][Designer: 3a][Programmer #1, #2, #3: 3a]

None of the team members were professionally active outside the company.[Analyst #1, #2: 6][Designer: 6][Programmer #1, #2, #3: 6] Analyst #1 cited cost and Programmer #3 cited lack of time as their reasons.

The team as a whole revealed no uniform understanding of corporate goals and objectives and how OOSD helps accomplish them. This could probably be expected since there was no high level champion to point out the benefits of OOSD to corporate direction. Designer and Programmer #3 seemed definitely aware of OOSD's application to the goals and objectives.[Designer: 7][Programmer #3: 7] Programmer #2 was "aware of the goals and objectives to some degree" and OOSD's application. Analysts #1 and #2 and Programmer #1 were not sure the goals and objectives addressed software with Programmer #1 stating OOSD was not used "to meet the goals and objectives."[Analysts #1, #2: 7][Programmer #1: 7]

The user team had no involvement in choosing an appropriate software project with which to gain OOSD experience.[Analyst #2: 8][Designer: 8][Programmer #1: 8] As Champion #1 pointed out, "there really wasn't a choice" to be made.[Champion #1: 15b]

Since there was not a champion facilitating the introduction of OOSD, the user team could not use communication channels to keep him informed of the project status. Any problems had to be communicated via the

standard chain of command. The interviewees had mixed feelings about the communication channels. The analysts had a different chain of command than did the other team members. They had no problems being heard.[Analysts #1, #2: 10] Analyst #2 felt the other team members had communication problems in that they were reluctant to use their chain of command when problems occurred. Designer felt communications were fine.[Designer: 9] Programmer #1 said past communications were poor but are improving.[Programmer #1: 10] Programmer #2 felt things get lost and Programmer #3 was pessimistic about using the channels.[Programmer #2, #3: 9, 10]

The perceptions of upper management support were diverse and opposed. Within the analysts' chain of command support was viewed as good.[Analyst #1: 11a] Within in the other member's chain of command, support was rated between poor and good. Analyst #2 felt the others' management was unsupportive through most of the effort.[Analyst #2: Inserted Question I, para. 2: 11a] Designer feels current support is strong but was weak at the project's start. She feels the project getting into trouble brought attention and needed support.[Designer: 11a] In contrast, Programmer #1 felt the support was good at first and became "sour" when the project got into trouble.[Programmer #1: 11a] Programmers #2 and #3 feel the support vacillates and negatively affects mor-

ale.[Programmer #2, #3: 11a] These differing perceptions do not seem to have been well managed.

There was no workable reuse library being established for multi-project use.[User Support: 16] The team members were not involved in creating one.

User Team Personality

Seven of the interviewees were asked to judge the user team on the percentage of members who possessed four characteristics. Table 6.1 represents their answers. Respondents are identified only by a number. This is the one question that many interviewees were uneasy about. Therefore, it is intentional that no connection is made to the interview answers in Appendix B.

Person 6 responses were viewed as too large an anomaly and were not considered part of the standard deviation calculations.

Most team members seemed to posses the desired characteristics. Person 6 scores the team low in works well and communicates well. Person 7 scores the team at 50 percent in all but self-confidence. With the rather high standard deviations, there is plenty of room to improve team composition. The comments under Pooled Data in appendix B suggest the problems and missed schedules negatively impacted self-confidence. Though persistence was generally high, some people did quit due to the long hours.

Table 6.1

User Team Personality Data

Legend

A Works well with others
B Communicates well with others
C Self-confidence
D Persistence

Person	A	B	C	D
1	.90	.90	1.00	.98
2	1.00	.95	1.00	1.00
3	.75	.95	.75	.85
4	1.00	.80	1.00	1.00
5	.60	.70	.60	.80
6	.19	.33	.60	.80
7	.50	.50	.75	.50
Mean	.71	.73	.81	.85
Mean minus #6	.79	.80	.85	.86
Std. Dev. minus #6	.21	.18	.17	.19

OOA Team

One analyst had seven previous projects performing requirements analysis. The other's first effort as an analyst was this project.[Analyst #1, #2: 14]

Since the object paradigm was not applied in the analysis phase, five of the six component operations were not performed. In fact, there was no OOA Team.

The only operation performed was the revisiting of the analysis phase as a function of an iterative life cycle approach. However, as previously stated, this was not a planned occurrence.

The firm has looked at OOA but has no current plans for its implementation.[Manager: 13]

OOD Team

As stated earlier, there was no design team separate from the programming team. A company contact stressed the AFCAP project was based on OOD. Accepting this tenet, this paper investigates the presence of the OOD team attributes and operations even though the OOD team and OOP team are the same in the case study.

Designer was experienced in systems design.[Designer: 3a, 3b] Given the development work structure where programmers do their own design, this paper will assume any experience in programming translates into design experience also.

Designer had no input into choosing vendor OOD

training and tools.[Designer: 14b] The firm generally uses its own training organization to conduct any training.[Champion #1: 10a]

The degree of learning OOD required by the framework was not achieved for this project. The training received merely "g[illegible] [them] going in the right direction." The training did not adequately train them for OOD.[Designer: 14b, 15]

Analysis to design communications took place but the object paradigm was of limited, if any, benefit in facilitating this communication.[Designer: 16a] The analysts were unfamiliar with the object paradigm and as such none of the benefits of improved analysis to design transition were realized.[Designer: 16c, 16b]

The designers did communicate with the users and analysts but the discussion of objects was limited.[Designer: 17] The benefit of common terminology offered by the paradigm was therefore unrealized.

The designers produced an OOD specification but it contained traditional design approaches as well. Designer cites lack of understanding and time constraints as causes for the less than ideal design product.[Designer: 14a, 15]

Using an iterative life cycle approach, the designers did revisit the design process; but, again, it was not by choice. Not understanding the benefit of this

type of life cycle and planning its use caused difficulty for Designer. She dislikes the approach.[Designer: 13]

OOP Team

All programmers sampled had previous programming experience on other projects.[Programmer #1, #2, #3: 3a, 3b] Question 3c of the programmer question set was designed to do a rudimentary test of the programmers' understanding and past application of basic structured programming principles. Of the two sampled, there was an understanding of the principles but not a strict application in practice.[Programmer #2, #3: 3c]

None of the programmers sampled had any involvement in choosing the language or vendor training and tools for use in the project. The language used was mandated and the company's own training organization handled OOSD education.[Programmer #1, #2: 14] Programmer #3 was brought to the effort well after the beginning of the project which precluded his involvement.[Programmer #3: 14]

There was no permanent maintenance advisor for the programmers to work with. This operation was performed to a minimum degree though, in that a person skilled in code maintenance was brought in to judge the programmers' work.[Champion #1: 8c]

The operations learn OOP method and programming language were performed but not to the degree required of

the framework. All programmers sampled applied the object-oriented principles.[Programmer #1, #2, #3: 16] However, a common feeling was the training received did not adequately prepare them for the task.[Programmer #1, #2: 15][Programmer #3: 14, 15]

The programmers did communicate with the analysts but since the analysts were unfamiliar with the object paradigm, this communication was in terms other than objects.[Programmer #1, #2, #3: 19a] Programmer #3 seems to not understand the broad reach of the object paradigm and its applicability to systems analysis. Even though this communication took place it did not occur when necessary at times. The analysts reported developers ignoring some requirements or adding non-required features. This caused problems for the customers in the end product.[Analyst #1: 4, para. 2][Analyst #2: 20, para. 2] Not using an OOA method prevented benefitting from common terminology in communications and led to expected problems. Requirements were interpreted one way by the analysts and a different way by programmers.[Analyst #1: 20]

Maintenance Advisor

As stated above there was no permanent maintenance advisor on the studied project's user team. Someone outside the team was brought in to perform the operation of judging the maintainability of the code but the full

benefits of this activity were probably limited due to the short time involved.[Champion #1: 8c]

Technology Resource Center

This firm's equivalent to the TRC provides tools and consulting support to all development projects. This support organization is funded by the various projects paying for services rendered.[User Support: 1]

The requirement of easy access to the TRC is questionable here. Having to pay for any support received may reduce frivolous requests but it also may reduce valid ones.

There is no evidence this TRC understands the basic principles and purpose of OOSD. The person sampled seemed to view C++ as **the** means of achieving software engineering goals and the object paradigm as an obstacle. [User Support: 2] Nearly every time I asked a question using the term OOSD, the answer I got in return was expressed in terms of C++.

OOSD did not enter the firm through the TRC, therefore, they did not seek a champion.[User Support: 3, 4]

User Team Support Unit

User Support was confident in his ability to provide resources and said he had plenty of contacts to help. [User Support: 5]

User Support reiterated that training of company

personnel is accomplished almost exclusively by the firm's own training organization. There was no choice to be made in who provides the training.[User Support: 6]

User Support felt decision maker support was lacking towards OOSD. Again, his answer was expressed using C++.[User Support: 15a, 15b] Funding came from each supported project and was not directly controlled by the decision makers.[User Support: 15c.]

The Object Reuse Library

The company provided a library of "reusable" components which was designed for use company wide, not just at this site. As stated earlier the user team found the components did not fit the needs of the AFCAP project. User Support criticized the library further.[User Support: 16, para. 1]

My contact stated User Support was involved in establishing a reuse library, however, I found this was not so. User Support reported reuse efforts were not impressive and "there is no real reuse library here." [User Support: Inserted Question III, para. 2: 16, para. 1] His attitude towards reuse was less than positive.

The prospects of this firm establishing a reuse library as the framework envisions seem dim. If this remains true, one of the major benefits of OOSD will be lost.

The Vendor

The fact that this company had the resources to develop its own tools and training programs seemed to restrict its looking at outside vendors. Since the firm provided the services of the vendor, this paper holds the firm should mirror the vendor framework component.

There seemed to be no object-oriented development tools provided. Manager felt lack of tools hindered productivity. [Programmer #3: 15, para. 2][Manager: 11a]

Every person interviewed at mid-level management or below felt the training was inadequate.[Programmer #1, #2: 15][Programmer #3: 14, 15][Manager: 5c, 5d][User Support: 2, para. 2][Champion #1: 15] This training failure seemed to have a direct impact in the project getting into trouble.[Champion #1: 11, para. 2][Programmer #1: 15] This conflicts with Senior Manager's view that the "training staff . . . is quite expert" and "the training was probably adequate."[Senior Manager: 4c]

The user team's needs were far more than the three day class provided. There were no indications the training was adapted to specific needs. Designer indicated that those wanting training received a standard presentation.[Designer: 14b]

When problems arose, the user team chose not to seek the training organization's help. They sought guidance from another person at their site.[Champion #1: Inserted

Question V]

The Decision Makers

Only one person was sampled who matched the decision maker profile. This one person's answers carry much weight, however, in that he was one of the most senior managers at the site.

This person definitely had the authority to control his assigned resources. He was in charge of product development of the business unit which owned the project under study.[Senior Manager: 1a]

Senior Manager stated a keen awareness of software's importance to his organization. He pointed out software was the determining factor in cost and delivery time for his products.[Senior Manager: 2]

Senior Manager seemed versed in the overall benefits of OOSD, but he was far from convinced they were true. His answers indicated he is a careful observer of the activities in his organization. His observations of the OOSD efforts led him to say he had not "seen evidence to support the claims."[Senior Manager: 4a, paras. 1 & 3: 5]

When trying to determine his understanding of the short-term and long-term costs of OOSD he responded with his view that the paradigm chosen and the associated costs are not the most important factors in software development. He went on to explain that starting a

critical project with a paradigm in which no one had experience was not a very wise decision.[Senior Manager: 11] This idea will be taken up more fully in the discussion on the appropriate software project below.

No champion support was given due to the absence of any true champions.

Senior Manager's description of the support he gave the AFCAP project indicated strong commitment to the effort. He pointed out some severe measures he took to correct the problems may not have looked like support to some. He normally uses the management hierarchy to relay support but with this project he took the time to meet with the project managers and "keep up the support as needed."[Senior Manager: 7a, 9a, 9b] As stated earlier, some at lower levels did not feel senior management support was good.

Corporate Goals and Objectives

There were corporate goals and objectives established but the data does not clearly indicate whether or not they fit the framework component profile.

There were differing opinions on whether or not the goals and objectives addressed software. Three people stated they now address software needs.[Champion #2: 5a] [Designer: 7][Manager: Inserted Question IV] The analysts stated the goals and objectives did not address

software, however, their answers should be qualified since they are not in the same chain of command as the others sampled.[Analyst #1, #2: 7] Senior Manager said they indirectly address software.[Senior Manager: 3b] It is very interesting that opinions on goals and objectives are so diverse. Software development is the major production activity at this site.[Senior Manager: 2][Analyst #2: Inserted Question II][Designer: Inserted Question: I]

The goals and objectives seemed to require the benefits which OOSD can provide. Some that saw OOSD as helping goal attainment reported the following benefits: improved quality; reduced maintenance time and costs. [Designer: 7][Programmer #2, #3: 7]

The framework requirement that goals and objectives be specific and easily understood was a function of communications in the case study. Senior Manager says every person under him is informed of the "fairly specific" objectives.[Senior Manager: 3a] As stated earlier though, not everyone at the staff levels were clear on the goals and objectives. Manager reports they were communicated "sort of indirectly."[Manager: 10b]

Appropriate Software Project

The one fact most apparent in this case study is this firm did not use an appropriate software project to judge the feasibility of OOSD.

A key requirement is the appropriate software project be non-critical to immediate corporate success. To say the case study project was critical is an understatement. Completion of other revenue generating projects were dependent on the object-oriented product. Three levels of management were replaced because of the problems. The delayed market delivery substantially reduced expected profits.[Senior Manager: 4a-para. 2, 7a, 7b][Analyst #2: 11a]

The project was similar in size and purpose to previous projects in that it was a version upgrade.[Designer: 16c-para. 2]

The project did not provide ample opportunity to practice OOSD methods. As several people noted, the user team received minimum OOSD training and went straight into development of the critical project.[Programmer #1: 15][Programmer #2: Inserted Question I][Senior Manager: 4c] Manager's views directly support the need for a non-critical project to practice on.[Manager: 5d] Senior Manager indirectly supports this argument. He points out they did not have enough experience with OOSD before starting.[Senior Manager: 4c, 4a-para. 2, 11-para. 2] Senior Manager did point out that they had an organization which evaluates and practices with new technologies but there is no apparent method for adequately transferring the knowledge.[Senior Manager: Inserted Question I]

There was no champion to study and adjust OOSD management methods but it seems the management team benefitted from the opportunity. Manager mentioned a "retrospective report" which outlined the project history.[Manager: 12a] This should provide valuable insight should future OOSD projects occur.

The Management Team

Since there was no champion to lead this pioneering project, the task fell to the management team. The group comparable to the framework component is seen by this paper as the management hierarchy below Senior Manager. Manager is the only member from this hierarchy who volunteered to be interviewed.

Manager had a cautiously positive attitude about OOSD and felt the technology would "in the long run . . . be good for the company."[Manager: 1a]

He expressed a basic understanding of OOSD and its differences over traditional methodologies.[Manager: 5a]

He had some understanding of OOSD's application to corporate goals and objectives and indicated involvement in communicating this information to the team. A more direct communication approach may have overcome some of the previously mentioned confusion in the team.[Manager: 6a, 10b]

Close inspection of Manager's answers indicates he

did monitor the attitudes, perceptions, and progress of the user team.[Manager: 5d, 6c, 7a]

He was brought in as a replacement for one of the managers relieved of duties. As such he was unavailable to perform the other champion operations which are required at project beginning.

Communication Channels

Most of what can be determined about the communication channels is revealed in the analysis above. This suggested communication needed improving. Items of note were: goals and objectives not adequately communicated; perceptions of training at staff levels were opposed to those at the senior level; inadequate analysis to design communications resulted in requirements to product discrepancies; communication to glean information from outside sources were almost nonexistent.

One other serious communication lapse was the reported lack of sharing of OOSD knowledge among team members at times.[Analyst #2: Inserted Question I][Programmer #3: 4a]

Summary of Findings

This section provides a brief summary of the more pertinent findings revealed in the above analysis. The information is organized under the headings Technology, Managerial, and Other Findings.

Technology

The firm developed the case study project with only an object-oriented design and object-oriented programming emphasis. While the omission of an OOA method is understandable due to the method's recent maturing, the benefits discussed in chapter III are obviously lost. The firm has no plans to integrate OOA into product development in the future.

The effort to implement OOD and OOP in such a critical product was bold but did not result in a pure object-oriented product. The severe time pressures led the developers to revert to traditional techniques for some parts of the system. They regretted this but felt there was no other means to meet the deadlines.

Some benefits were realized even with this first major OOSD effort. Improvements in maintenance and quality were mentioned by several.

Managerial

Inappropriate Software Project

If there is one key factor leading to the severe problems this project had, it would have to be the first time application of OOSD to a highly critical system. It was apparent through all discussions that developer inexperience with OOSD caused the majority of the problems. One could argue Senior Manager's addition of new people to the project [Senior Manager: 9a] disrupted the

teamwork and led to the missed schedule. This paper counters this argument in that the problems started well before this event.

Had the user team been given a chance to practice and gain needed OOSD experience on a non-critical project, many of the problems may have been avoided.

Inadequate Training

Another major factor contributing to the problems was the inadequate initial training in OOSD. The U.S. Air Force has a training team which teaches a programming language and its proper application at several sites around the world. This language is equivalent in difficulty to C++. The course is 152 hours long, offered only to experienced programmers, and provides only a starting point in experience.[1] The three days of training given to the user team seems totally inadequate for learning C++, let alone OOD and OOP. As the analysis revealed, they were behind almost from the start.

While confidence in the firm's training organization is expected, total dependence on it for all training neglects outside avenues for possibly better instruction.

Lack of Champion

The organizational structure at this firm did not seem conducive to a champion approach as envisioned by

---

[1] Direct experience - I supervised one team's training.

the framework. Project managers were in charge of product development but had to rely on systems engineers (analysts) from a different chain of command. The project managers were the ones responsible for paradigm selection.[Senior Manager: 11] A high level champion well versed in OOSD could probably better handle the conflicts of two different command levels. Very few of the champion activities proposed by the framework and supported by literature were adequately performed for this project.

A true champion may have better handled the user team dynamics, met their needs, and established a proper environment for OOSD utilization.

Other Findings

Reuse is the OOSD benefit most written about in literature. This organization has no obvious plans to seriously research its proper application. Their attempt to employ general libraries established at another company location was unsuccessful.

No one interviewed had a clear understanding of OOSD's applicability across the entire software life cycle. This can be expected to limit funding and research into the full benefits of OOSD.

CHAPTER VII

CLOSING DISCUSSIONS

This chapter concludes the thesis by discussing OOSD's chances for diffusion in the studied firm, the limitations of the case study research, and additional research areas.

Prognosis for Diffusion

If current conditions persist at the studied firm, the diffusion of OOSD can be safely predicted to never occur. Yes, the user team does prefer the technology. There are also those outside the studied project which favor OOSD.[Programmer #3: Inserted Question III][Manager: 7b] However, the serious problems encountered in the AFCAP project have cast a dark cloud over OOSD. Several people stressed the technology itself did not cause the problems, but no one can erase the negative impression cast by a late project and lost profits while OOSD was being used.

Senior Manager's interview provides the most condemning evidence against OOSD's diffusion. He has seen improvements in small projects using OOSD but only problems in large projects. He says if OOSD cannot be ap-

plied to large efforts "then it's not going to be worth much here."[Senior Manager: 4a-para. 3] He said he does not blame OOSD for the AFCAP project's problems, but the tone of the interview shows he feels OOSD to be too risky. To reverse Senior Manager's view of OOSD, a major project success would have to occur. This seems unlikely as this thesis found no evidence suggesting other program managers are willing to chance OOSD.

Limitations of the Case Study

One case study is not adequate for testing the framework's soundness or for suggesting its adjustment. Follow-on research of other companies is needed to compile a suitable database which will support or refute the framework. Needed is a broad spectrum of case studies. Some studies should be done in companies whose OOSD implementation environment closely resembles the framework. Others need to be done in firms with environments dissimilar to the framework. These further studies should target companies whose software is applied in different industries.

Since one sample limits the conclusions that can be made, this thesis argues for keeping the framework in its present form. Even with the limitations, the analysis and findings in chapter VI do imply that an appropriate software project and adequate vendor training are sound

framework principles.

Additional Research Areas

This section identifies two items for further research. One item was revealed in the case study interviews. The other is deduced from the nature of OOSD technology.

Business Unit Impact

One surprising revelation was the suggestion that organizations which restructure into semi-autonomous business units may actually deter the possibility of object reuse.

The interview with User Support found this company had traditionally transferred tools, software products, and other innovations freely among its many sites. The different sites evaluated the usefulness of these products to their own missions and applied them at will. The restructuring into business units has stopped this free technology transfer. If one business unit wishes to use or evaluate software developed by another business unit, it must first pay for it. User Support says this will effectively stop reuse in the company. [User Support: 16, Inserted Question III]

The above information suggests avenues of research outside the scope of this thesis. While business units may help large corporations be more competitive, will

they also neutralize the major advantage that large companies have - the ability to bring enormous capital and diverse expertise to bear on a competitive idea? Will making autonomous business units compete among themselves for capital and expertise actually stifle innovation causing missed opportunities for competitive advantage? The case study suggests even if OOSD was adopted, this company would not benefit from reuse. Are there other companies which have this problem? Other research is necessary to answer these and related questions.

OOSD and Market Share

Chapter III suggested that an intangible benefit of OOA is increased customer satisfaction. The object paradigm allows the use of OOA to express the problem domain in terms familiar to the domain experts - the customer. Carrying customer defined objects throughout the development effort allows the solution domain to more closely match the problem domain.

This thesis found no data to support using OOA via OOSD impacts market share one way or the other. Research into this area may reveal a positive correlation and help bolster OOSD usage.

Conclusion

This thesis has defined the object-oriented princi-

ples and an effective means to apply these principles through OOA, OOD, and OOP. It suggests that OOSD is a potentially useful software engineering technology with many promising benefits.

OOSD is not a technology that one person can evaluate and adequately judge its value to a company. OOSD's broad reach across an entire software development life cycle requires many people to be part of the adoption process and to provide input into the company's decision to implement or not. A proper environment needs to be established to effectively manage the adoption process. This thesis offers one such environment via its framework architecture.

The detailed case study of one company tested the presence of the framework components in an OOSD project. This one study was not sufficient to thoroughly evaluate the framework but it did start the process of building a database from which a judgment can be made. The case study did suggest some framework components were sound.

While the framework has yet to be proven, this thesis nonetheless offers it as a viable means to introduce OOSD into a corporation.

REFERENCES

Banker, Rajiv D., and Robert J. Kauffman. "Reuse and Productivity in Integrated Computer-Aided Software Engineering: An Empirical Study." MIS Quarterly (September 1991): 375-399.

Beudette, Neal E. "Object-Oriented Programming: Untangling the Software Mess." Industry Week, 5 March 1990, 49.

Beatty, Carol A. and John R. M. Gordon. "Preaching the Gospel: The Evangelists of New Technology." California Management Review (Spring 1991): 73-93.

Biggerstaff, Ted J. "Software Reuse: Is It Delivering?" In 1991 IEEE 13th International Conference On Software Engineering by the IEEE. Los Alamitos: IEEE Computer Society Press, 1991, 52-54.

Boehm, Barry W. Software Engineering Economics. New Jersey: Prentice-Hall, 1981.

Booch, Grady. Software Engineering with Ada. Menlo Park: The Benjamin/Cummings Publishing Company, 1983.

_____. Object-Oriented Design With Applications. Redwood City: The Benjamin/Cummings Publishing Company, 1991.

Bouldin, Barbara M. "Automate your software development with minimum pain," EDN, 7 June 1990, 111.

Brancheau, James C. and James C. Wetherbe, "Information Architectures: Methods and Practice," Information Processing and Management 22, no. 6 (1986): 453-63.

Brancheau, James C. and James C. Wetherbe. "Understanding Innovation Diffusion Helps Boost Acceptance Rates of New Technology", Chief Information Officer Journal, Fall 1989, 23-31.

Clarke, Dan. "Object Insider: Diskette frisbee isn't fun." Object Magazine, May/June 1992, 19.

Coad, Peter, and Edward Yourdon. Object-Oriented Analysis. Englewood Cliffs: Yourdon Press, 1991.

Cox, Brad J. "There Is a Silver Bullet." Byte, October 1990, 210.

Drucker, Peter F. Management: Tasks, Responsibilities, Practices. New York: Harper and Row, 1974.

Duff, Chuck, and Bob Howard. "Migration Patterns: Moving to object-oriented technology is more involved than simply buying a compiler." Byte, October 1990, 224.

Dykman, Charlene A., and Ruth Robbins. "Organizational Success Through Effective Systems Analysis," Journal of Systems Management 42 (July 1991): 6-8.

Fischer, Gerhard, Scott Henninger, and David Redmiles. "Cognitive Tools for Locating and Comprehending Software Objects for Reuse." In 1991 IEEE 13th International Conference On Software Engineering, by the IEEE. Los Alamitos: IEEE Computer Society Press, 1991, 318-328.

Flaatten, Per O., Donald J. McCubbrey, P. Declan O'Riordan, and Keith Burgess. Foundations of Business Systems. Orlando: Dryden Press, 1989.

Frenzel, Carroll W. Management of Information Technology. Boston: Boyd & Fraser Publishing Company, 1992.

Gabriel, Richard P., and Patrick Dussud. "Neoclassical object-oriented techniques: multimethods." Object Magazine, May/June 1992, 41.

Giegold, William C. Management By Objectives: A Self-Instructional Approach, vol. 1, Strategic Planning and the MBO Process. New York: McGraw-Hill, 1978.

Gray, Linda, James C. Brancheau, and Kenneth A. Kozar. "Evaluation, Environment, and Education: The 3E Framework for CASE Implementation." Software Engineering: Tools, Techniques, Practices, March/April 1992, 18.

Gupta, Ashok K., and David L. Wilemon. "Accelerating the Development of Technology-Base New Products." California Management Review 32 (Winter 1990): 24-45.

Hagedoorn, John. The Dynamic Analysis of Innovation and Diffusion. London: Pinters Publishers Limited, 1989.

Henderson-Sellers, Brian, and Julian M. Edwards. "The Object-Oriented Systems Life Cycle." Communications of the ACM 33 (September 1990): 142-159.

Herzog, John P. "People: The Critical Factor In Managing Change." Journal of Systems Management 42 (March 1991): 6-11.

Hodges, Parker. "A Relational Successor?" Datamation, 1 November 1989, 47.

Howard, William G., Jr., and Bruce R. Guile. Profiting from Innovation: The Report of the Three-Year Study from the National Academy of Engineering. New York: The Free Press, 1992.

Howell, Jane M., and Christopher A. Higgins. "Champions of Change: Identifying, Understanding, and Supporting Champions of Technological Innovations." Organizational Dynamics 19 (Summer 1990): 40-55.

Johnson, Ralph E., and Brian Foote. "Designing Reusable Classes." Journal of Object-Oriented Programming 1 (June/July 1988): 22-35.

Kellner, Marc I. "Non-Technological Issues in Software Engineering: Panel Session Overview." In 1991 IEEE 13th International Conference On Software Engineering, by the IEEE. Los Alamitos: IEEE Computer Society Press, 1991, 144-146.

Kinlaw, Dennis C. Developing Superior Work Teams. Lexington: Lexington Books, 1991.

Kolman, Joe. "Getting down to CASEs." Institutional Investor, August 1990, 119.

Korson, Tim, and John D. McGregor. "Understanding Object-Oriented: A Unifying Paradigm." Communications of the ACM 33 (September 1990) 40-60.

Litchfield, Charles A. "Vendor Training: A Question of Commitment to User Success." Journal of Library Administration 12, no. 2 (1990): 3-12.

Martin, James. "OOP Goes Beyond the Commonsense Meaning of 'Object'." PC Week, 11 September 1989, 76.

McClure, Carma L. Managing Software Development and Maintenance. New York: Van Nostrand Reinhold Company, 1981).

Meyer, Bertrand. Object-Oriented Software Construction. Hertfordshire, Great Britain: Prentice Hall International Ltd., 1988.

Moad, Jeff. "Cultural Barriers Slow Reusability," Datamation, 15 November 1989, 87.

Nachmias, David, and Chava Nachmias. Research Methods in the Social Sciences. New York: St. Martin's Press, 1981.

Norman, Ronald J., Gail F. Corbitt, Mark C. Butler, Donna D. McElroy. "CASE Technology Transfer: A Case Study of Unsuccessful Change." Journal of Systems Management 40 (May 1989): 33-39.

Norman, Ronald J. "Object-Oriented Systems Analysis: A Methodology for the 1990s." Journal of Systems Management 42 (July 1991): 32.

Preece, David A. Managing The Adoption Of New Technology. London: Routledge, 1989.

Puhr, Gretchin I., and David E. Monarchi. "Object-Oriented Analysis and Design: A Comparison of Methodologies, Techniques and Representations." Faculty Working Paper Series, College of Business and Administration, University of Colorado - Boulder, 1991.

Ralston, Anthony, and Edwin D. Reilly, Jr., eds. Encyclopedia of Computer Science and Engineering. New York: Van Nostrand Reinhold Company, 1983. S.v.
"UNIVAC I" by Michael M. Maynard.
"Software Maintenance," by Saul Rosen.
"Software Engineering," by C. V. Ramamoorthy and K. Siyan.

Schlumberger, Maurice. "Software Engineering Management." In 1991 IEEE 13th International Conference On Software Engineering, by the IEEE. Los Alamitos: IEEE Computer Society Press, 1991, 152-153.

Seidewitz, Ed. "General Object-Oriented Software Development: Background and Experience." The Journal of Systems and Software 9 (1989): 95-108.

Sommerville, Ian. "Object-Oriented Design: A Teenage Technology." In Software Engineering for Large Software Systems, ed. B. Kitchenham, 316. London and New York: Elsevier Science Publishers LTD, 1990.

Stefik, M., and D. Bobrow. "Object-Oriented Programming:

Themes and Variations." The AI Magazine, Winter 1986, 43.

Swander, Alice J. "Vendor Training: Establishing the Evaluation Process." Performance & Training 29, no. 4 (April 1990): 32-35.

Ten Dyke, R. P., and J. C. Kunz. "Object-Oriented Programming." IBM Systems Journal 28, no. 3 (1989): 465-478.

Tully, Colin. "A Failure of Management Nerve and Vision." In 1991 IEEE 13th International Conference On Software Engineering, by the IEEE. Los Alamitos: IEEE Computer Society Press, 1991, 154-155.

Verity, John W., and Evan I. Schwartz. "Software Made Simple." BusinessWeek, 30 September 1991, 92.

Wasserman, Anthony I. "Object Insider: Object-Oriented Thinking." Object Magazine, September/October 1991, 10.

Weber, Charles A., John R. Current, and W.C. Benton. "Vendor selection criteria and methods." European Journal of Operational Research 50, no. 1 (January 1991): 2-17.

Wirfs-Brock, Rebecca J. and Ralph E. Johnson. "Surveying Current Research in Object-Oriented Design." Communications of the ACM 33 (September 1990): 104-124.

Wishart, Nicole A., and Lynda M. Applegate. "The Prudential: Organizing For Technology Innovation." Harvard Business School Case Study. Boston : Harvard Business School, 1990.

APPENDIX A

CASE STUDY INTERVIEW QUESTION SETS

This appendix contains the interview questions categorized by question set type.

The Champion Question Set

1. How do feel about your firm using OOSD to develop software?

2. Do you see yourself as sort of the champion of OOSD in your area? or site?

3. What position do you hold in the organizational structure?

4. What is the management hierarchy above you?

5a. What do you feel are the major differences between objects and other ways of developing software?

b. How did you gain this knowledge of objects?

6a. What parts of the OOSD methodology do you feel help meet your corporate goals and objectives?
b. Have you found any time reductions?
c. Do you foresee any cost reductions?
d. Has OOA assisted customer relations as far as defining requirements faster and/or easier?
e. Any other benefits?

7a. Did you begin the process of converting your firm to OOSD or did someone else seek you out to lead the effort?
b. If someone else sought you, what is their position in the firm?
c. Were any problems encountered while going to OOSD?

8a. Are you using a team concept to implement OOSD?
b. Did you assemble or oversee the assembling of this team?
c. Does this team consist of software analysts, software designers, programmers, and at least one maintenance person?

9. What is the attitude of the team members toward OOSD?

10a. To what extent where you involved in choosing OOSD methods and tools?

b. Were the team members and (TRC equivalent) included in the selection process?

11. To what extent where you involved in training the team members in these methods and tools?

12a. Do you feel it's important that the team members understand how OOSD helps meet corporate goals and objectives?
b. If so, was the team educated on these facts?

13. How do you view your short term costs associated with implementing OOSD?

14a. Do you feel those who guide the corporation firmly support the OOSD effort?
b. What efforts have you taken to communicate this support to those involved with the OOSD effort?

c. How do you view communication channels from the top? (Easy letting them know problems and getting feedback?)

15a. Where you given the freedom to choose an appropriate software project with which to implement OOSD?
b. To what extent did you involve the user team and (Technology Resource Center equivalent) in this choice?
c. How critical is the project to immediate corporate success?

16a. Do you see yourself as the project leader?
b. To what extent do you actively monitor the attitudes, perceptions, and progress of the team as they implement OOSD?

17. How do you feel about your rapport with others in the firm? (i.e. are there any closed doors, do you feel you can communicate easily with those in other departments, etc..)

18. Are you associated with any professional technology oriented organizations outside your firm?

19a. Would you consider your professional contacts outside your firm to be many, few, or none?

b. Of these, do you have many, few, or none which are technology oriented?

Modified Champion Question Set

1a. How do feel about your firm using OOSD to develop software? (Is it a viable technology?)
b. How do you see the future of OOSD at your company?

2a. In what ways did you try to champion OOSD here?
b. Did you begin the process of converting your firm to OOSD or did someone else seek you out to lead the effort?
c. If someone else sought you, what is their position in the firm?

3. What position do you hold in the organizational structure?

4a. Do you feel there needs to be a higher level proponent of OOSD?
b. Do you know of anyone at a higher level who likes the technology?

5a. Has OOSD helped meet any of your corporate goals and objectives on any projects that you are aware of? (Any time reductions? Any cost reductions?)

b. Are you aware of OOA being used to assist customer relations as far as defining requirements faster and/or easier?

6. How many projects are using OOSD here?

7. Are you currently on a project using OOSD?

8. Are you associated with any professional technology oriented organizations outside your firm?

9. Would you consider your professional contacts outside your firm to be many, few, or none? Of these, do you have many, few, or none which are technology oriented?

The Analyst Question Set

1a. How familiar are you with object-oriented methodologies?
b. What do you feel are the main principles of OOSD?

2a. How do you feel about object oriented software development as a viable development technique? (Good

for your company?)

b. Do you use any object-oriented techniques in establishing system requirements?

c. Have any benefits been realized with OOSD? (cost reductions, time, etc.)

3a. How many software development efforts have you been involved with?

b. How confident are you in your ability to use OOSD to successfully complete the chosen software project?

4a. On the AFCAP project do you work with the other developers as a team or on an as needed basis?

b. Does your office location affect your working relationship in any way?

5. In your view, what percentage of your fellow AFCAP team members posses the following characteristics?

a. Works well with others.

b. Communicates well with others.

c. Self-Confidence.

d. Persistence.

6. How many professional technologically oriented organizations are you associated with outside your firm?

7. How does OOSD help meet your firm's goals and objectives?

8. What type of input did you have in choosing the software project that your team was to develop using OOSD?

9. As you use OOSD, do you feel opinions you may have about the methodology are being heard by the right people?

10. Do you feel communications channels are always open for you to express any problems that occur? (ie. no problem getting the right people informed of the problems and keeping them informed of the progress?)

11a. How do you feel about the support from the top?

b. Anyone ever tell you how (the decision makers) feel about your efforts?

12a. Have you worked with (TRC equivalent) in setting up an object reuse library?
b. Are you satisfied with the efforts in this area?

13a. What type of life-cycle approach are you using?
b. Are you using an iterative/fountain life-cycle approach (may need to explain to interviewee) with OOA, OOD, and OOP or are you sticking with traditional transition phases?

14. How many software projects have you been involved in performing requirements analysis?

15. What's your overall impression of OOA? In what ways is it better than previous methods you have used?

16. Were you involved in choosing a vendor to provide OOA training and tools? Who did you work with in doing so?

17. With the training you received, are you confident you can produce a quality object oriented requirements document?

18. Have OOA techniques helped you communicate better with the end-users as opposed to the past methods you've used? (ie. have you been able to express their requirements more in their terms via objects?)

19. To what extent do you feel your requirements specification has helped or hindered the design team as opposed to past methods?

20. Do you feel your analysis team and the design team communicate and work well with each other? (is there agreement on what the project requirements are?)

The Designer Question Set

1a. How familiar are you with object-oriented methodologies?
b. What do you feel are the main principles/elements of OOSD?

2a. How do you feel about object oriented software development as a viable development technique? (Good for your company?)

b. Did you use any object-oriented techniques in doing the system design of the AFCAP project?
c. Have any benefits been realized with OOSD? (cost reductions, time, etc)

3a. How many software development efforts have you been involved with?
b. How many software projects have you been involved in performing software design?

4a. On the AFCAP project do you work with the other developers as a team or on an as needed basis?
b. Does your office location affect your working relationship in any way?

5. In your view, what percentage of your fellow AFCAP team members posses the following characteristics?

a. Works well with others.
b. Communicates well with others.
c. Self-Confidence.
d. Persistence.

6. How many professional technologically oriented organizations are you associated with outside your firm?

7. How does OOSD help meet your firm's goals and objectives?

8. What type of input did you have in choosing the software project that your team was to develop using OOSD?

9. As you use OOSD, do you feel opinions you may have about the methodology are being heard by the right people in management?

10. Do you feel communications channels are always open for you to express any problems that occur? (ie no problem getting the right people informed of the problems and keeping them informed of the progress?)

11a. How do you feel about the support from the top?
b. Anyone ever tell you how any senior leaders feel about your efforts?

12a. Have you worked with (TRC equivalent) in setting up a object reuse library?
b. Are you satisfied with the efforts in this area?

13. What type of life-cycle approach are you using? In other words, are you using an iterative/fountain life-cycle approach (may need to explain to interviewee) with OOA, OOD, and OOP or are you sticking with traditional transition phases?

OOD Team Specific

14a. What type of object-oriented design approach did you use - pure or hybrid?
b. What was the extent of your involvement in choosing a vendor to provide OOD training and tools?
15. With the training you received, are you confident you can produce quality object-oriented logical and physical design documents?

16a. Have OOD techniques helped you communicate better with the analysts as opposed to the past methods you've used?
b. Is it easier to move from analysis to design?
c. If no to either question, why do you think communications or analysis to design transitions have not improved?

17. Do you get together/communicate with the analysts and users to insure you have the correct interpretation of the objects and their interaction?

18. Do you:
a. reuse object classes/modules? If so, is the reuse library helpful and easy to use?
b. make use of inheritance and polymorphism?
c. design classes to be reused?

19. To what extent do you feel your object oriented design has helped or hindered the programming team as opposed to past methods?

20. Do you feel your design team and the programming team communicate and work well with each other? (is there agreement on what is to be implemented?)

The Programmer Question Set

1a. How familiar are you with object-oriented methodologies?
b. What do you feel are the main principles/elements of OOSD?

2a. How do you feel about object oriented software development as a viable development technique? (Good for your company?)
b. Have any benefits been realized with OOSD? (cost reductions, time, etc)

3a. How many software development efforts have you been involved with?
b. How many software projects have you been involved in as a programmer?
c. How well do the terms modular, tightly cohesive, and loosely coupled apply to your past coding efforts?

4a. On the AFCAP project do you work with the other developers as a team or on an as needed basis?
b. Does your office location affect your working relationship in any way?

5. In your view, what percentage of your fellow AFCAP team members posses the following characteristics?
a. Works well with others.
b. Communicates well with others.
c. Self-Confidence.
d. Persistence.

6. How many professional technologically oriented organizations are you associated with outside your firm?

7. How does OOSD help meet your firm's goals and objectives?

8. What type of input did you have in choosing the software project that your team was to develop using OOSD?

9. As you use OOSD, do you feel opinions you may have about the methodology are being heard by the right people in management?

10. Do you feel communications channels are always open for you to express any problems that occur? (ie no problem getting the right people informed of the problems and keeping them informed of the progress?)

11a. How do you feel about the support from the top?
b. Anyone ever tell you how any senior leaders feel about your efforts?

12a. Have you worked with (TRC equivalent) in setting up a object reuse library?

b. Are you satisfied with the efforts in this area?

13. What type of life-cycle approach are you using? In other words, are you using an iterative/fountain life-cycle approach (may need to explain to interviewee) with OOA, OOD, and OOP or are you sticking with traditional transition phases?

OOP Team Specific

14. What was the extent of your involvement in choosing a vendor to provide OOP training and the programming language to be used on this project?

15. With the training you received, are you confident you can produce quality object oriented code?

16. Using the terms extensively, moderately, very little, or none, to what extent are you using the following elements of object-oriented programming? (You will need to get their view/definition of each of the terms so you can be sure you're talking the same language)

a. Objects.
b. Abstraction.
c. Inheritance.
d. Encapsulation.
e. Polymorphism.

17a. Do you use the reuse library to help reduce the amount of code you generate?
b. What's your opinion of reusability (think it's going to have much impact)?
c. What's your opinion of your reuse library?

18. Do OOP techniques help you work and communicate better with the maintenance advisor in order to produce more maintainable code?

19a. Have OOP techniques helped you communicate better with the analysts and designers as opposed to the past methods you've used?
b. Is it easier to move from design to coding?
c. If no to either question, why do you think communications and/or design-to-coding transitions have not improved?

20. What's your opinion of the quality of code produced by OOP? Is it better, same as before, or worse?

The Mid-Level Manager Question Set

1a. How do you feel about your firm using OOSD to develop software?
b. How do see the future of OOSD here?

2. Is there any one or two people you see as championing OOSD here?

3. What position do you hold in the organizational structure?

4a. What position does your supervisor hold?

b. What do you feel the attitude is of those above you toward new technologies? (eager to try them, let others try them first, won't try them unless they've been used several other projects and succeeded, try them if forced to)

5a. What do you feel are the major differences between objects and other ways of developing software?
b. How did you gain this knowledge of objects?
c. Do you think the team received adequate training in OOSD?
d. How would you have made it better?

6a. What parts of the OOSD methodology do you feel help meet your corporate goals and objectives?
b. Have you found any time reductions?
c. Do you foresee any cost reductions?
d. Has OOA assisted customer relations as far as defining requirements faster and/or easier?
e. Any other benefits?

7a. What do you feel is the attitude of the team members toward OOSD?
b. Do you have any idea/feeling of what others outside the AFCAP project think about OOSD?

8. In your view, what percentage of the AFCAP team members posses the following characteristics?

a. Works well with others.
b. Communicates well with others.
c. Self-Confidence.
d. Persistence.

9a. Were any of the above characteristics considerations in choosing the development team for AFCAP?

b. What was the most important consideration?

10a. Do you feel it's important that the team members understand how OOSD helps meet corporate goals and objectives?
b. If so, was the team educated on these facts?

11a. How do you view your short term costs associated with implementing OOSD?
b. Do you feel those above you were aware of these costs? If not, why not?

12a. Do you feel those who guide the corporation firmly support the OOSD effort?
b. What efforts have you taken to communicate this support to those involved with the OOSD effort?
c. How do you view communication channels from the top? Easy letting them know problems and getting feedback?

13. Are you familiar with object-oriented analysis and its benefits to software development?

14. Are you associated with any professional technology oriented organizations outside your firm?

15a. Would you consider your professional contacts outside your firm to be many, few, or none?
b. Of these, do you have many, few, or none which are technology oriented?

The Technology Resource Center Question Set

1. Give me an idea of how your support function works here.

2. How does your center help meet your firms goals and objectives?

3. Did your center originate the research into OOSD for possible use by the firm?

4. Did your center seek and/or find someone to champion OOSD?

The User Team Support Function

5. Are you confident your training has given you the skills to find and provide the computer resources

and training necessary for developing quality software systems?

6. How familiar are you with all available resources for performing OOSD?

7. What was the extent of the champion and analysts' involvement in helping you choose a vendor and resources for OOA training?

8. What was the extent of the champion and designers' involvement in helping you choose a vendor and resources for OOD training?

9. What was the extent of the champion and programmers' involvement in helping you choose a vendor and the language for OOP training?

10. Did you substantiate each vendor's past track record in providing the kind of training and tools needed?

11a. Did you get feedback from the team members on vendor instructors?
b. Did they consider the instructors were properly skilled?

12. Did you request any of the vendors to adapt their tools or training to meet any special needs you had? Did they do so?

13a. Did you need any post-purchase support on any of the training or tools you purchased?
b. Was support fast and effective?

14. Do you feel you were able to provide resources to the user team at the times needed?

15a. Do you feel you get appropriate support from top management?
b. Has support been actively expressed from the decision makers?
c. Do you feel your group gets enough money to do what is required?

The Object Reuse Library

16. Has a library of reusable objects and classes been established?

17a. Is the library easy to use by those needing it?
b. Is access fairly easy and non-cumbersome?

c. Are search tools designed to find modules which match or are similar to designs sought?

18a. Do you have a control system in place to ensure modified modules are tracked?
b. Have you had problems with old bugs returning?

19a. Is the library system automated?
b. Is it accessible via networks or other remote systems?

20a. Do you consider yourself skilled in configuration management practices?
b. How did you gain your knowledge?

21. Were members of the user team consulted in establishing the reuse library to meet their needs?

22a. Did you seek out current reuse library technology in setting your standards for storage and retrieval?
b. Does your system have a meaningful classification of components?
c. Does your system document component design and original purpose?

The Decision Maker Question Set

1a. What is your position in the organization? (or: In what way do you influence the direction of the organization?)
b. What is your background prior to obtaining your present position? (software, hardware, management, finance...?)

2. What are your feelings on software's role in your organization? How important is it?

3a. Do you feel you have specific and easily understood corporate goals and objectives which provide the direction for your organization?
b. Do these goals and objectives address the importance of software?
c. Are these goals and objectives divided into long-term (or strategic) areas and short-term (or tactical) areas with regards to software?

4a. What do you see are the benefits of OOSD?
b. Do you see the technology as the cause of the problems in the AFCAP project?

c. Do you think the AFCAP team received adequate training in OO techniques?

5. Do you feel OOSD can improve your business opportunities and help achieve your corporate goals and objectives? If so, is this communicated to those involved with the OOSD effort?

6. Do you see anyone in your firm as a champion of OOSD?

7a. Do you see yourself as supportive of the people on the AFCAP project?
b. Are you satisfied with their efforts so far?

8. Do you actively communicate your support (or dislike) for OOSD to others?

9a. How actively do you communicate your support to the people on the AFCAP project and others in your firm?
b. By what means do you ensure they are aware of your support?

10. Have there been any high level disputes over OOSD where you had to run interference/defend the champion? If so, what were the general reasons for the dispute?

11. What are your feelings on the high short term costs of OOSD versus the long term payoffs?

APPENDIX B

CASE STUDY INTERVIEW DATA

This appendix contains the answers to the interview questions found in appendix A. No interviewee names are given in order to ensure confidentiality as much as possible. The acronym AFCAP is the mask assigned to signify the name of the software project studied. Any other masking of identifying information will be so noted with square [] brackets. Square brackets are also used to insert any clarifying information.

The interview answers are presented under the heading of the corresponding question set. Additional questions asked which are not part of the question set are signified by **INSERTED QUESTION**. Since all answers are quotes they are single spaced.

Prior to the start of any interview, each participant was told of his/her right not to answer any question and that she/he could terminate the interview at any time. Each was reminded that no names would be published with the responses and that the company identity would remain anonymous.

I recorded the original responses via hand written notes. I had time between interviews to add from memory any details I may have missed. The responses below were generated from these notes at the earliest opportunity.

The Champion Question Set

Champion #1

1. Object-oriented design and object-oriented programming are better than top-down functional methods to develop software. The maintenance is improved. There has been a problem using C++ on this project. C++ has required far more RAM than the previous version did. This has affected the existing base of hardware by requiring newer machines to run the system effectively.

2. When the project started, I was sort of seen as the champion. There was one other supporter for using OOSD. Now, there are several advocates of the methodology. There really is no one person seen as THE champion of the methodology.

INSERTED QUESTION I: Is there anyone in the hierarchy above project levels who is seen as an OOSD advocate? No one that I know of.

INSERTED QUESTION II: Do you have a feel for how other project groups see the technology? Other groups are wary of using OOSD because this project was late. That and I don't think they like C++. The object paradigm is not the reason we were late. One reason the project was late is that product management wanted more capabilities than were feasible in the time given. They greatly underestimated the time needed. We didn't have enough training time to learn object-oriented techniques before we had to produce usable code. We didn't have time to practice. Management also wanted to mix object-oriented techniques with a functional approach and I think that was a big mistake. The finished system has several areas that need an object-oriented approach but don't have and this hurts the systems as far as maintenance goes.

3. [Staff level position]

4. Looking at this site only, there are 4 successive levels above me.

5a. [See 1]

5b. I read a book by Brad Cox; I took a one week class in C++ on site; and I played around with the technology on my own.

6a. The goals and objectives address maintenance and quality. OOSD does produce more easily maintained soft-

ware. Maintenance costs are reduced. The quality is better.

6b. Being late with the project, naturally there were no time reductions in original development. But, now we're seeing time reductions in debugging and fixes from problems found in field tests.

We probably could have realized other benefits but we just didn't do as much front end design as we should have. Managers wanted the code finished and weren't as concerned with the design phase. Same old development problems.

6c. [see 6a]

6d. The analysts deal with the old ways, functional based. I don't see where the object paradigm could help them.

6e. OOSD emulates the real world better. Functional software development doesn't represent the real world as well.

7a. I'm not aware of anyone else pushing OOSD at the same time we started this project. [The company] had other projects using OOSD at other locations. I think there were other projects here using OOSD but I'm unsure. I didn't interact with any of the other projects.

7b. No one sought me out.

7c. There were some political problems. At the time there were other project groups doing similar types of efforts as ours and they went over our heads and tried to keep us from using object-oriented techniques.

8a. Yes.

8b. I didn't have any real influence on building the team; I was asked my opinion on a few things.

8c. The original team had analysts, we call them system engineers, and the job of designing and programming are combined. What I mean is the person who is to program a piece of the system also designs it. As far as maintenance, we didn't have a permanent member. We brought in someone from another site and for one week he evaluated the maintainability of our work. I felt he was a bit brief, though.

INSERTED QUESTION III: What do you think about having

someone full time in a maintenance advisory mode? I think that would be a good idea.

9. At the beginning of the project, some were ambivalent and others resisted. Of the 10 original programming members, 2 didn't want it at all, 2 were all for it, and the rest had the attitude of I'll do it if I have to. Now, all present members say the object paradigm is better.

10a. There weren't really any useful code support tools at the time. I really had no involvement in choosing methodology training. [The company] has its own training program and they gave us a 1 day course in object-oriented design and a 1 week course in C++. I don't think the training we received was sufficient for the work we had to do.

INSERTED QUESTION IV: Does [the company's] training organization do any type of follow-on evaluation of their training? They had us fill out an evaluation at the end of the training but they haven't done any evaluation after we've had a chance to use the training.

INSERTED QUESTION V: Did you have any questions or problems that required you to contact [the company's] training people for help? We had some problems but we didn't go back to the training people. There was a local [company] person who had experience in C++ and OOSD and we took our questions to him.

10b. No.

11. I wasn't really involved with the training but if anyone on the team had any problems, they would come to me first.

12a. I feel it's important, but I'm not sure everyone knows how OOSD meets corporate goals and objectives.

12b. Through memos and department meetings.

INSERTED QUESTION VI: Do you try to make the leadership aware of how OOSD helps meet the goal and objectives? I'm not sure if my supervisor is passing up this information or not.

13. The short term costs are very high especially in training. Going to an object-oriented approach requires a change in mind set. As you learn OOSD you are climbing a mountain and it's not obvious what you need to do. Once you're over the top though, I think its obvious how

to use objects.

14a. I don't feel there's much support for OOSD at the top. I think the lateness of the product put us in a negative light. Management is not too sure of OOSD right now.

14b. They know.

14c. I don't feel the doors are too open. You're sort of expected to use the chain of command when you have a problem. I get little or no feedback from senior management.

15a. I was on the AFCAP project team and some of us wanted to go with OOSD.

15b. There really wasn't a choice of projects to apply the technology. [see 9 for member attitudes]

15c. The project turned out to be more critical than we realized. Other projects depended on our work. It turned out we didn't have enough machine resources at first and we started to have problems. We were a low resource priority and sometimes our compiles would take all day to complete. As we began to be late, management realized we needed more resources and we got them.

16a. No; the department head is the project leader. He has other projects going on that aren't object-oriented.

16b. N/A [Not appropriate for his level]

17. I don't deal with others here. There's just not enough time to do that. Lack of time has kept me from getting the object-oriented message out.

18. Yes, one. IEEE.

19a. Not very many.

19b. I don't get information from other professionals.

Modified Champion Question Set

Champion #2.

1a. It's viable and I think it should be used.

1b. I think it will have to wait. It's been very slow

to catch on with people here.

2a. I started on my own using the technology for my own small projects. The department head wanted me to spread the knowledge so I started by sponsoring lecture series on C++. We had about 3 to 6 lectures a year. I was able to bring in some big name people in OOSD and C++. I established a support agreement with [the company's] object-oriented people at [another location]. I talked to the former site director about the technology but he wasn't too interested. I was able to give a presentation to a vice-president in [another location] and he did seem interested. As I've talked about OOSD, I've found camps for and against it. Every time I give a presentation to project directors, some will be all for it and some will be adamantly against it.

2b. [See first two lines of 2a]

3. I'm part of the technical staff.

4a. Here, managers don't really push what techniques are to be used on a project. I'm not sure having those higher pushing a technology would work. I think those in the lower level development organizations must be convinced to go with OOSD if it's going to work. One thing about technologies, what is smiled upon today may be frowned on tomorrow.

Even when certain techniques are good for improving quality, quality goes by when things are going to be late. I don't see any strict standards for quality. When a project is late, whatever it takes to get it out the door is what goes.

4b. That vice-president was interested, but there is no one actually pushing it that I know of.

5a. Goals and objectives change and they haven't always addressed software development. For about a year now they have focussed on software. On a previous project I worked on, there was a positive improvement in quality. Changes were easier to make with OOSD and this cut time requirements.

5b. I'm not aware of anyone using object-oriented analysis here. The systems engineers [analysts] in my area are looking at it but no action has been taken yet.

6. Three that I know of.

7. No.

8. Yes. ACM and IEEE.

9. Few. I occasionally talk to them; I don't get much information.

The Analyst Question Set

Analyst #1.

1. I'm not very familiar. What I've heard is changes that you want to make are far easier with OOSD. I don't feel this will be much use; programmers aren't going to like many changes. They will feel too much chaos is involved with mandated changes in design.

2a. I haven't seen where OOSD has made much difference, yet. I know there was a long lead time in the developers coming up to speed in learning it.

2b. We didn't use any OO techniques in building our requirements.

2c. I really haven't seen any benefits.

3a. 5 efforts, 1 as an analyst.

3b. N/A

4a. I felt I was part of the team. When the project got into trouble the team really unified and everyone really pulled together.

4b. My location had no affect on our work. But, there were communication problems at times. When I needed to talk to someone about a certain piece [project module or code] it was hard at times to determine who was assigned to it. At times no one felt responsible for various pieces and it made my job harder. Sometimes the developers wouldn't consult us and would ignore some of the requirements. Some things [required elements] didn't make it to the end product.

5a-d. [See Table 6.1 in chapter VI]

6. 0. I was member of one in the past, just wasn't worth the money.

7. The goals and objectives really address making more money. I don't really feel they address software.

Management does emphasize it would like software developed faster and at cheaper cost.

8. N/A

9. N/A

10. My chain of command is different from the development team's. I work close with the marketing people in the business and am therefore separate from the product development people in supervisory levels. Even so, I do work as a team member with development in establishing the system requirements. As far as communications go, I was able to voice my frustrations when trouble occurred. I feel I was heard but since my chain of command is different, I can't say if the others' communications were hindered.

11a. Within my supervisory chain, the support is fine. As I worked with the development team, I felt support was OK there too.

11b. Within my chain, you don't really get much feedback. If you don't get feedback, that generally means you're doing a good job.

12a. No.

12b. N/A

13a. We used the iterative life-cycle out of necessity not because of the application of the object paradigm. The waterfall life-cycle just doesn't work in complex systems. By waterfall I mean where you're supposed to come up with requirements that everyone agrees on and then they're set in stone. In complex systems as you go along requirements change; something may have been left out or the customer wants something altered.

13b. We are looking at better ways to come up with requirements; but we're not looking at any OO analysis methods.

Inserted Question I: Are you aware of anyone championing the object paradigm here? No.

14. See 3a.

[Analysts did not use OOA, therefore questions 15 - 19 were not asked]

20. There's not much trouble with communications. Our

styles are different in that we have different world views. We develop the requirements by focusing on the customer. In designing the system the developers don't focus as well on the customer. They are more concerned with the underlying architecture and getting something to work. There were a few problems, though. There's always trouble in trying to translate the real world to the software world. We've found after production completion, that our requirements were not totally complete. Actions which we would interpret as "obvious" were somehow interpreted differently by the designers and showed up in the final product as not quite the actions we intended. We have to work to correct these problems in the future.

Inserted Question II: Do you have any idea how the customers view the product? We are currently testing the product at 6 controlled release sites. So far the feedback is generally good. For some reason some have complained about the differences from the past versions of the product. I can't quite explain their reasoning since this is supposed to be a major upgrade with several new enhancements.

Analyst #2.

1. I'm not very familiar with it. My idea of it is it's a structured way to program.

2. I can't give you an opinion due to my lack of knowledge.

3a. 12-15 software development efforts.

3b. N/A

4a. I now have other responsibilities besides the AFCAP project. Originally, as the requirements were being developed I was totally dedicated the AFCAP project full time. I was definitely a team member.

4b. My location now doesn't interfere with my working relationships. At the beginning of project I was in [another state] and transferred here shortly after start of the project.

5a-d. [See Table 6.1 in chapter VI]
[Side issue during discussion of this question] The team had a lot of trust and worked well together. A problem was there wasn't any formal communications established where team members actively communicated new techniques and learning with others.

Inserted Question I: Were OO methodologies a problem to the team? I feel they [designers/programmers] were forced into the language. I feel the OO learning wasn't shared with others. Those that learned something new or useful didn't always share that with those that could've benefitted. Some understood the technology far more than others.

[Additional comments on persistence] The project got in trouble and got behind. Upper management responded to the problem by discouraging anyone from taking vacations and wanting a lot of overtime. Many were working 70 hours a week for two years. I feel management's actions reduced the group's normal persistence. I think the group felt management didn't trust them. No one quit from this though. New management came in later and reversed previous decisions. They encouraged people to take vacations and get away from the work for a while.

6. 1 in the past; 0 now.

7. Goals and objectives don't really address software. I'm not very familiar with the goals and objectives.

Inserted Question II: What percentage of this site's productivity is in software? 90%

8. None

9. N/A. [Didn't use OOSD]

10. My chain of command goes through [another state's] office. I have no real communication problems, things work fine. I feel others in the team had communication problems. When the project got into trouble, no one wanted to say anything. No one wanted to get management involved when things started to get bad even though I thought someone should.

11a. There seemed to be a general lack of information at the upper levels. The team didn't want to elevate problems. I don't think they were confident in the upper level support. At times I don't feel management was very interested in the project. They are very interested now because of the money impact. This project was critical to other products being sold. After the project got into trouble, it got managements attention. Now that the project is doing well, I think management is proud of the team.

11b. [Not asked]

12a. No.

12b. N/A.

13a&b. For the original version of the project we developed the requirements in [another state] and sent them here for designing and coding. After the requirements were sent here we didn't get them back for updating with missed or new requirements. These original requirements didn't have everything the customer needed, though. After I was transferred here we went to an iterative cycle out of necessity. The requirements were changing a lot. I'm not sure if OOSD caused this [use of iterative cycle] or not. Management wants requirements written in stone but in practice everyone knows you can't do this; requirements change during development. The original manager didn't like it [changing requirements] but the new manager does. He says he likes having up-to-date requirements.

14. 8.

[Since OOA was not used, questions 15 - 19 were not asked.]

20. The design teams changed over time. This is a problem now because I was given additional responsibilities as the project matured. These responsibilities have nothing to do with the AFCAP effort. There was no formal communications established which kept us informed of these changes. This caused problems for us in that when requirements changed or we needed to talk to someone about a module, we didn't always know who to call and it was hard finding out at times. We would view project parts differently at times than the developers and it wasn't easy at times to figure out who was working on which aspect or feature of the project.

Sometimes the developers decided to put in a requirement on their own. They didn't consult us. In field testing the product, these new additions have caused a few problems for the customer.

The Designer Question Set

Designer.

1a&b. Data abstraction, use of objects - where data and operations are combined into one entity.

2a. I feel it's viable now. I wasn't sure at the beginning of this project because of some problems with people coming up to speed using it.

2b. Yes; both in design and programming. The way we work here is the developers do the design and programming of their pieces of the project. I had more of a hand in the beginning with the design as far as establishing the original system architecture.

2c. This being the first big project using the paradigm, we haven't seen many benefits yet. I feel the maintainability has improved a lot. It's easier to make changes without causing other problems.

3. Six. All six.

4a. The original design was a team effort. As the project was divided into pieces and pieces assigned to people we worked some as a team. The low level design was mainly individual work; there wasn't much interaction. The output [coded product] could have been more of a team effort I feel. Management kept pushing us to get the product finished and we didn't take the time to make sure everything meshed together well. While there weren't major problems, I felt the final product could have turned out better if there was more time for teamwork.

4b. No. We all work in the same area.

5. First, let me say the team changed over time. Some of the original people left and new people were brought in when we had some trouble. The original core of people worked well together but the group dynamics were interrupted by the new people. For both the old and new people this was a factor in some communications problems.

5a-d. [See Table 6.1 in chapter VI]

6. None.

7. Management has laid out their goals and objectives and they do specifically address software issues. The only benefit we've realized so far is improved quality. Our quality is a lot better using the object paradigm.

Inserted Question I: What percentage of this site's total productivity is concerned with software development? 95%

8. None.

9. Yes. My concerns are being heard and I feel management cares about what I have to say. I think management is confident in what I and the others are doing.

10. Yes. Answers to my concerns generally always come back to me.

11a. Now I think it's strong from the top. I feel it was weak in the beginning. When we got into trouble and started slipping schedules we were able to get a lot of help and support.

We underestimated the learning curve for the object paradigm and got into trouble as we attempted to implement it. We also had problems with lack of hardware resources. We needed more computer time but couldn't get it. Sometimes our compiles would take all day to complete due to our low priority. Because our project was so critical, after we got into trouble, we got all the resources and support we needed.

11b. [Not asked]

12a. We have libraries that are supposed to be for reuse and in the beginning we tried to use them but the original library modules we used had too many bugs or they were just too slow for what we needed. The original libraries weren't that good.

12b. I feel there is only a limited application for a reuse library with our project. The AFCAP system is unique and most modules are just too specific to really apply reuse.

13. We used an iterative approach out of necessity, not because of the paradigm. I don't see the iterative lifecycle as a plus, though. It just takes too much time. It's very time consuming to have to go back to requirements when something is not identified up front or when the customer wants something else. We were constantly updating the requirements with the AFCAP project and it got to be too much.

14a. I'd say it was a hybrid approach. We used some techniques from older methods. I didn't really understand OOD at first. We were able to get help and input from other experienced people.

14b. I had no involvement in choosing who trained us. Our training people bring in people occasionally to present lectures on different topics. People can attend as they want. As far as OOD, they set up a three day

class on OOD. We learned what we could in three days. They at least got us started in the right direction.

15. Now, yes. When we started it was a struggle. Like I said, the training got us going in the right direction but did not give us everything we needed. I think trial & error had more impact in our learning than the training. Our design and documentation could have been a good deal better in the AFCAP system but time constraints kept problems in.

16a. As far as interaction with the systems engineers [analysts], we talked objects with them but I'm not too sure they really understood what we were saying. It has helped communications among the developers in that we focus on objects when we talk about the system. It helps in that we're not continually focusing on each function and small part.

16b. There's no real improvement.

16c. The system engineers don't understand objects.

[Interviewee added comment] My understanding of how the AFCAP system operates is far clearer with Version 3 over Version 2.
[Version 2 was implemented with traditional development methods and a non-object-oriented language.]

17. Yes. We spent quite a bit of time gaining an understanding of the requirements. But, like I said, we really didn't talk objects that much.

18a. Yes. We used a few library modules for some pieces. We usually made minor modifications. We didn't put our modified modules back into the library. I wouldn't say it was easy to use.

18b. Inheritance, yes. Polymorphism, yes.

18c. I design for reuse now; not at first.

19 & 20. In the AFCAP project, each piece was designed and coded by the same person. In the future we may separate the functions.

The Programmer Question Set

Programmer #1.

1a&b. I think encapsulation is a key part of OOSD. In the old ways of programming you used a linear approach and just wrote code as you thought about it. C++ forces me to think about what I need ahead of time. What I mean is I have to look at how modules interact before I can write my code.

2a. I think it's a good technique and it will probably stay around.

2b. The code is more maintainable. The older methods produced code which was cumbersome and hard to update.

3a. 3

3b. 3

3c. [Not asked]

4a. Definitely a team effort.

4b. No. As you saw, we are all close together.

5a-d. [See Table 6.1 in chapter VI]

6. None.

7. I guess it has helped in making the product more maintainable. We really didn't use the technology to meet the goals and objectives. The corporate goals and objectives haven't really been spelled out for me.

8. None.

9. I really don't have any opinions that I want elevated other than I don't want to go back to using C.

10. Channels are fairly open; they are improving. In the past, communications were poor. [The company] did a survey in the past which identified communications as poor here.

11a. Originally, we had good support. When the project got behind schedule the support turned sour. Some people on the project were forced to retire. Now we have pretty good support again.

11b. No.

12a. [Not asked]

12b. [Not asked]

13. [Not asked. The life-cycle being used has been established.]

14. No involvement.

15. After the decision was made to implement this project with the object paradigm, we were behind schedule from almost day 1. The project finally came on line almost 1 1/2 years late. [The company] has its own training organization and they gave us a 1 day class on OOD and a 2 day class on C++. After practicing on a few sample programs, we started programming the AFCAP project. We really had inadequate training to get up to speed. We ended up trying to develop a critical product too fast with a new technology. The schedule was just unrealistic in this environment.

Now, I'm confident I can produce quality code.

16a. Extensively.

b. Extensively.

c. Moderately.

d. Extensively.

e. Some. A bit less than moderately. I don't really like to use polymorphism. I feel it makes it hard for people who'll look at my code in the future to understand. When I use polymorphism and then come back to my code later, I have trouble figuring out what my own code is doing.

17a. Reuse was not attempted too much. Time constraints prevented identifying reusable modules. There was a lot of duplicate code in this project.

17b. I'm really not sure.

17c. The libraries that we used had too many bugs.

18. N/A. [No maintenance advisor]

19a. The analysts don't know the technology or the terminology used; so no it hasn't.

19b. No.

19c. The programmers and designers are one in the same here.

20. The quality seems to be a lot better. I really don't know why though.

INSERTED QUESTION I: Do you talk to others outside your work area about OOSD? No.

INSERTED QUESTION II. Do you have any idea if others outside your team like the technology or not? I have no idea.

Programmer #2.

1a. I've done some object-oriented design work in the past.

1b. The use of polymorphism and inheritance.

2a. If there are successful projects here, it will fly. We've had some problems with the AFCAP project. It wasn't a success at first, but we're turning it into a success. It's over budget. With AFCAP, OOD wasn't used as much as it could have been. There's a mixture of some good object-oriented design and pieces with no OOD. Early on I think the design was good but as the project became late, OOD was pretty much abandoned. In programming, people tended to regress from C++ to using straight C.

INSERTED QUESTION I: What do you think caused these problems? I don't feel there was enough training. There just wasn't a good understanding of the object paradigm early on. There was a one day course in OOD and it really took a few months to get the feel of it. With AFCAP there was no real chance to practice what was learned. Things were left in the code which should have been altered or thrown away. I feel people should get a chance to practice with something new. This was too critical a project to learn as you go.

2b. Yes. I found it easy to adjust and fix problems. Enhancements are easy to make without changing the interfaces. The object paradigm makes it easy to add features. I was brought to this project from another which was using the object paradigm and the benefits there were more pronounced. That project didn't have the pressures of this one.

[ASIDE COMMENTS] I'm primarily responsible for maintenance on this project. We're currently field testing the product and I fix the bugs that are found. I've been on the project about a year.

3a. 12

3b. 12

3c. My past code was very modular. It was tightly coupled though. Cohesiveness is always the goal, but wasn't always reached.

4a. This was not nearly the team I would have liked it to be. Sometimes different ideas clashed and instead of resolving things, some would just go off on their own.

4b. [Not asked]

5a-d. [See Table 6.1 in chapter VI]

6. None.

7. I'm aware of the goals and objectives to some degree. OOSD helps with software maintenance. Costs are being reduced; time to service code is reduced. Overall, the code is more maintainable. I don't feel the higher ups are doing well at stating the goals and objectives as far as OOSD goes. The efforts toward the object paradigm came from the grass roots. I feel the goals which address software were accepted by senior leadership because of the efforts of lower level workers.

8. [Not asked. No one had a choice.]

9. Upper level supervisors aren't very technology oriented. My immediate supervisor always respects my opinions but they haven't gone any higher than him.

10. People listen to my ideas but things get lost. There's no real formal communications mechanism.

11a. When things are going well we get good support. When we get into trouble we get plenty of negative feedback. When we got into trouble, management seemed to make a point of publicizing their concerns that we weren't going to get the job done. This negatively affected morale.

11b. I've received a few awards and pats on the back on occasions; I've never talked with the senior leaders.

12a. No, I didn't know of his efforts with reuse.

12b. [The company's] reuse library is not very useful. It allows us to get prototypes going very fast but the code doesn't perform very well in the actual product. The code is just too general. Reuse within the AFCAP project has gone well. Modules from other components are used but we don't use modules from outside or other projects.

13. [Not asked. Fact has been established.]

14. I wasn't involved. Use of C++ was mandated by others and [the company] provided the training.

15. I was when I was fresh out of training. My first experience using the paradigm showed me otherwise, though. I had quite a bit of learning still to do.

INSERTED QUESTION II: Does [the company's] training organization evaluate the effectiveness of their training? They have a questionnaire at the end of training but they don't evaluate the training after it's been used for real.

16a. Extensively.

b. Moderately. It wasn't used as much as I feel it should. People tended to regress to straight C.

c. Moderately.

d. Moderately. Could have been used more.

e. Very little. Should have been used a lot more.

17a. No, not the large libraries. In the AFCAP specific libraries I do.

b. [Not asked.]

c. [See 12b.]

18. N/A. [No maintenance advisor]

19a. No. The analysts don't seem to be familiar with object-oriented terminology and so using it is a hinderance. They really don't see what is possible with using objects.

b. It seems easier. We really don't have good design documentation with an object orientation.

20. Quality is high with the right design. As I said before, some pieces were designed using older methods. Quality is definitely better than before.

Programmer #3.

1a. I'm fairly familiar with it.

1b. The abstract data types and the object user interfaces such as those found in Smalltalk.

2a. OOSD is not the best solution for all problems. You must identify up front if the problem matches the paradigm. I feel OOSD will be of limited application here because of [the company's] devotion to UNIX. C++ is not a perfect language either. There are limitations to the paradigm. It is hard to share objects in large systems especially in parallel systems where objects can't be shared easily across processors.

2b. Yes. Reusability within the project; extendibility [he agreed his definition matched mine of inheritance]; reduction in time to code; and adding features seems a bit easier.

3a. 5
3b. 5

3c. My past code was modular, more tightly coupled than now, and highly cohesive.

INSERTED QUESTION I: Was your transition to OOSD easy or difficult? I had no problems moving into the object realm.

4a. Both. Sometimes we get to work as a team and other times I feel left out. Time crunches were a big factor in keeping people apart, that and the large amounts of data we had to pour through. I could generally get assistance when I needed it but at times I couldn't.

4b. Location is not a problem.

5a-d. [See Table 6.1 in chapter VI]

6. None. I don't have time to read other material. [The company] keeps us up to date on technology.

7. Maintenance is improved. New features can be added with minimum code additions. It's not clear to me that any start up costs were saved.

8. N/A. [He came to project after its start.]

9. It's not clear to me if my opinions are heard or not. There are too many management levels. I hardly ever get feedback on what I say.

10. The channels are open but you don't want to use them if you don't have to. If you elevate too many problems, you may give an improper view of things. I see somewhat of an adversarial view between management and us; this tends to limit some communication.

11a. Sometimes the support is lacking; there seems to be turf battles at times.

11b. No one from higher management has ever mentioned my specific efforts. My direct supervisor does commend and comment on my work.

12a. There have been some small efforts toward reusability.

12b. The libraries we used early on were insufficient. The code was not good grade and the libraries weren't well implemented; not easy to use. I'm not familiar with the current state of any reusability effort outside the AFCAP project.

13. [Not asked]

14. After I came to the project, I arranged for a class to help those of us inexperienced with OOSD, but the class was so basic that only 4 or 5 of us were new enough to receive any benefit. [The company's] training people didn't train our team as a group. Most of the C++ and object-oriented learning came as on-the-job-training and from books.

15. The training we received was not adequate to do any serious work with. Even the books we used didn't adequately address the problems we had to face in this project.

Now I feel experienced enough to program quality code. I do feel I need more training in object-oriented design addressing classes and hierarchies. I would like to have tools to help create objects more quickly, too.

16a. Extensively.

b. Extensively.

c. Extensively. I feel that I and others use inheritance maybe too much. It shortens the code but at times I feel creating a new class might be more beneficial.

d. Extensively.

e. Very little. I see increasing use in the near future.

17a. Within the project there are some reusable modules available which are used if possible.

17b. I think reuse is beneficial and is the way to go in the future. With the AFCAP system to get maximum benefit of reuse would probably require changing the architecture. You would need the resources to change the existing code to make it reusable and this would affect a large majority of the existing code.

17c. Not enough code is actually usable and the libraries aren't easy to use. There are no browsing tools to help find what you need.

18. [Not asked]

19a. We don't communicate with the analysts using object-oriented terms. I don't see where object terms would be beneficial to customers. Objects are for constructing the internal workings of a system.

19b. At first it was difficult because of the massive amount of code that the design called for. It was very tedious to generate the proper classes.

19c. Analysts aren't familiar with OOSD. [see also 19a]

20. Quality has improved.

INSERTED QUESTION II: Do you talk to others outside your work area about OOSD? Yes. A lot of project groups have shown interest in the past. There's not as much interest now; people have either accepted it or rejected it. I think the problems we encountered probably caused this.

INSERTED QUESTION III: Do you have any idea if others outside your team like the technology or not? All the software developers I know like it; they're all for it. Upper management has been opposed to it; they feel it's too risky.

Mid-Level Manager Question Set

Manager.

1a. In general it causes technical discontinuity. You stop doing things the way you're accustomed and have to learn a new methodology and this causes problems. The object paradigm will give us good results but these will come at a price. There is a steep learning curve; learning slows things down. In the long run it will be good for the company, but in the short run it's like starting over.

1b. The technology has a black-eye here because of the problems encountered with the project. Some see object oriented development as the cause but that is not necessarily so. Its use may grow but not as fast now. I see isolated uses in the near future.

2. There are a few in management ranks who like it and are seeking ways to use it. Its use comes in mainly from the staff level.

INSERTED QUESTION I: How effective are the managers in getting OOSD accepted? They have the authority to use it in new projects if they see fit.

INSERTED QUESTION II: How many projects outside the AFCAP project are using OOSD? 4 or 5 here.

3. [Answer omitted due to identifying information.]

4a. [Answer omitted due to identifying information.]

4b. They are willing to try new technologies if there is proof they work. They won't be ones to jump in first.

There has been a change in leadership since this project started but both leaderships had the same characteristic towards new technologies.

5a. I'm not an object-oriented expert. The major difference is that OOSD is a different way of thinking about a problem. You don't view systems development like you would in a traditional functional approach. I think the biggest feature is information hiding. Another good use is in windowing user interfaces where you establish icons as objects and build other objects around the icons.

5b. A two day class here and reading books.

5c. The learning curve is bigger than people realized.

I don't think they got adequate training.

5d. I feel adequate training is 6 months of prototyping using object-oriented techniques knowing you will throw away your results. This gives you a chance to learn what you should without worrying about a production deadline. This group went straight into a critical project. A lot of good people struggled with this in the beginning. Now they're proficient.

INSERT QUESTION III: Do those above you know your feelings about the training? I know my boss does. I don't know about those above him.

6a. Information hiding. This makes problems easier to fix without affecting other components. Easier fixes enhance maintenance and raise quality.

6b. Not yet. This is the first effort with this product. I feel the first effort using OOSD actually slows down the development.

6c. We'll see improvements in some areas of the project not others. Some pieces of the product didn't use object-oriented techniques. Some of the developers were worried about time crunches and used older methodologies to develop their code.

6d. N/A

6e. Reuse is not a big effort here. The developers have found the needs are so application specific that they just didn't get much use from existing libraries. In fact for the general modules they did hope to use, the reused modules actually degraded performance.

7a. It's a good motivation tool. People like to learn something new. Everyone in the team prefers OOSD and they wouldn't want to go back to the older methods.

7b. Outside this project, those who have used it like it. Others have unjustly blamed the technology entirely for the problems the AFCAP project has had.

8a-d. [See Table 6.1 in chapter VI]

9a. Get along with others.

9b. The number one consideration was getting people with a "can do" attitude.

10a. Yes.

10b. Through meetings and other communications the team was shown sort of indirectly how the goals and objectives were to be met. The primary goals which were focused on were increased productivity and reduced faults per non-comment lines of code.

INSERTED QUESTION IV: Do the goals and objectives specifically address software? Now they do; they didn't in the past.

11a. High. It takes a lot of training and the learning curve is very high. Getting to a productive stage is also hindered by a lack of needed OO development tools.

11b. No, they weren't told. The people initially pushing the technology didn't talk about the costs. That's only natural because I think there's always the fear if the negatives are discussed then you might not get the go ahead.

12a. The upper management here doesn't really feel the technology is sound due to the problems the AFCAP project had. They were supportive in the beginning but not very supportive now. We're going to try and change that with our retrospective report. This is a report to upper management on the problems and pluses of the project. We will show how the technology was not the main reason for the troubles encountered.

12b. [Not asked]

12c. Communication is reasonable. There are no real problems getting information up and down the channels.

13. There was an effort here at one time toward gaining knowledge on OOA but no one I know really found much use for it. The work load now prevents looking at it any time soon.

14. No.

15a. Few.

15b. A handful may be technically oriented.

The Technology Resource Center Question Set

User Support

1. I acquire implementations of C++ platforms; answer

questions on C++; provide a consulting role on C++ and existing objects which exist in the libraries. I help make C++ development tools and maintain the C++ libraries. I see my department as a loose collection of tools, with each unit responsible for their set of tools. We provide support to any project who needs us, we don't belong to any specific project. Our funding for existence comes from each of the projects. They pay us out of their funding for our assistance.

2. The software goals are to reduce faults per lines of code, increase robustness, and increase reliability. Using C++ does this. To do it right, you need to start with C++ as a better C for projects. You don't want to change the paradigm at first, you've got to get people use to the language. The fault rate goes down just with using the language.

Going object-oriented is an entirely new mind set. A 3 day class like they give around here isn't going to give you all the knowledge you need. As far as object-oriented techniques go, you need to see if there are any natural applications for using objects. I think people can go overboard with objects resulting in performance penalties. Calling a bit an object and having associated operations on a bit is ridiculous.

3. No. [Someone else] introduced OOSD here but at the time there were no projects at the stage where implementation was possible.

4. No.

User Team Support Specific

5. Yes. I don't have any problems with that. I have good contacts at other company locations that assist me with whatever I need.

6. I'm very familiar. Training is provided by a separate company training organization. I do bring in object-oriented people occasionally as they're needed, but this is rare. I don't think people are seeking help when they need it.

[Since I had established most of the training was company provided, I did not ask questions 7 through 13]

INSERTED QUESTION I: Is there anyone assigned in your company to research the object paradigm and its application? Not here. There is an organization in [another company location] which does.

INSERTED QUESTION II: Does anyone here communicate with them about their research? Not really. The ideas coming from there are very abstract and not really applicable here yet.

14. Yes. We do customer satisfaction surveys to determine whether or not needs were met. As I said before, they are paying customers and we want to keep them happy.

15a. As you go up the chain you'll find people who have used the older technologies. They are very skeptical of newer technologies. For them to accept C++, they will have to see increased profits and decreased costs. Some projects fell behind schedule and management blamed C++. It seems that minor problems at low levels get blown out of proportion at upper management levels. C++ definitely has a black eye.

15b. Not from upper management but yes from my boss and his boss.

15c. [Projects are their customers; money comes from there. See 1 and 14.]

Object Reuse Library Specific

16. There is no real reuse library here. The [another company location] center has C++ developers who built a library of stock classes for use company wide. This library was just too general, though. Classes were too large, too slow, and too dependent on other components in the library.

In the past, there was a kind of informal library system made of libraries put together by the different company locations. It was managed from a central location in [another site]. There was a free exchange of modules by any location who needed code developed at another site. But I think free exchange is going away. [The company] has reorganized into autonomous business units. Each business unit is responsible for its own resources and funding now. If one business unit wants to use a module developed by a different business unit then it has to pay for it.

INSERTED QUESTION III: What do you think this is going to do for reuse in your company? I don't think it's going to work. You're going to start having duplicate work going on and a waste of resources. No one is going to want to spend money on a module of code that they aren't sure may do what they want or they may have to modify a bit.

Efforts here with reuse haven't been too effective. There are problems with reuse. It costs money to track classes. You may spend as much time looking for a class as if you wrote it from scratch. I think developers lack trust in library modules; they're afraid of introducing errors. I think the efforts in reuse here are more salvaging, finding bits and pieces, than actual module reuse.

[Since there is no workable reuse library or dedicated work to establish one, questions 17 through 22 were not asked.]

The Decision Maker Question Set

Senior Manager

[I felt this person's answers would have more weight if more identifying information was left in the replies than is evident in the other answers. I asked the senior manager to review these answers and give approval to leaving this information in. Approval received.]

1a. I am in charge of all product development for one of [the company's] business units. I have people here at this site as well as in two other states. I have 1300 people under me.

1b. Hardware, circuit design. I became a manager in 1972 and was given jobs of increasing responsibility up to my current position.

2a&b. Software is the lifeblood of what we do here. In 1969 software was 2 percent of this location's efforts. Now, 70 percent of our people here are involved with software development. We have no products which are not software intensive. Software is the gating interest in cost and time to market for our products.

3a. I have a written set of objectives which are to guide our people in their efforts. I personally send out a copy to every person under me. The objectives which everyone receives is a fairly specific, 4-5 page document.

3b. Yes and No. What we have are project commitment

objectives written for project guidance a whole. Software is not specifically addressed. We do address concerns such as reducing faults per line of code and indirectly address software quality.

3c. Yes; our strategic objectives target areas such as revenue growth and process improvement and are less specific for the long term. Our tactical objectives are naturally more specific in how we plan to accomplish our goals in the short term.

4a. I've heard the claims of increased productivity, increased modularity, and reduced errors; I'm not convinced of these claims however.

I think some of our people here have confused using C++ as an automatic use of the object paradigm. The AFCAP project was implemented using C++ and the people thought they were using object-oriented techniques but they didn't seem well prepared for the new methodology. The project was almost 2 years late. This product was very critical in that other projects were dependent on it and had to be held up because of its problems. I'm hard pressed to say AFCAP was a good example of object paradigm use.

We do have a better success story with the TOOL project but it was a smaller effort to develop a tool set. It didn't require as much personnel and resources. I feel if the object paradigm is good only for small projects then it's not going to be worth much here. Our commercial products are large efforts and generally interface to other products. As far as I'm concerned, the jury is still out on just how beneficial the object paradigm really is. I haven't seen evidence to support the claims.

4b. I don't specifically blame the object paradigm for the AFCAP problems; there were other factors. I do know the object-oriented nature of the project seemed to confuse people during the code inspection process, however.

4c. [The company] has its own research and training staff which is quite expert in the object paradigm and C++. I feel the training was probably adequate. We have enough expertise and resources available to overcome most training problems. I feel the AFCAP people, in the beginning, thought they understood object techniques and what was required but the reality is soon after they started the project they had problems. Training alone isn't going to give you the development expertise needed

for such a large project as this. The object paradigm was new to most every member of the original team and as such, probably shouldn't have been used in this instance.

5. The jury is still out. As an observation, no other project manager in 3 years has chosen to use object-oriented techniques; but, these projects are mainly follow-on efforts and the original architecture was not developed using the paradigm. No new start-up projects have chosen it either.

6. There are some champions. There are some in our tool development unit which are OO experts and are proponents.

7a. Yes, I am supportive, some may have differing views on this. The AFCAP is a critical product and was in desperate need of completion. I ended up having to change 3 levels of management to help correct some problems. These changes had nothing to do with the object-oriented techniques, the job just wasn't getting done.

7b. Well, they're through the valley of death and they seem to be doing fine now. They have a new release due within the next year and I'll be looking to see if they are on time and under budget. To answer your question, I am not satisfied overall with the project. The project came in at double the time and cost and we're not going to be as profitable as projected.

8. N/A

9a. As far as my actively communicating, people know what their commitments are, I don't need to stay continually involved. On the AFCAP project, people put in long hours to meet their commitments. As they got into trouble I heard they needed more time and they received it; but schedules were still missed. We supported them with more people, resources, and paid them over-time. Adding new people to an ongoing effort can create social problems and, when this happened, we brought in people to smooth these problems and help motivate and encourage the team. It's sometimes hard to encourage people and make them see the importance of project completion without being seen as beating up on them. As a whole, I think we were very supportive of the project.

9b. Those things [normal support actions] are handled by the project managers. They meet regularly with their people and keep up with the needs and solve the problems. When action and support is needed at my level, I give it. During the height of the AFCAP problems I met once a week with their managers to monitor the progress and keep up

the support as needed.

10. N/A

11. Whatever design paradigm is used can be an aid to development but I feel that's second order in importance. There are two important considerations with any project: 1) did the project manager make a wise decision in making the development commitment (i.e. schedule, budget, promised functionality, etc)? 2) what is the expertise of the team?

If you make a bad decision and start a project in a hole, you're dead in the water. We don't dictate to project managers which development methods to use. It's their responsibility to analyze what is needed and to make the appropriate decisions. In the case of the AFCAP project I think the older technologies should have been used. It was just too critical a project to start with new techniques. While the team was composed of experienced developers, they were not experts in the use of object-oriented techniques. It seems the promises of the object paradigm probably led to an over-estimate of their productivity.

INSERTED QUESTION I: In the AFCAP case, the developers went almost straight into production after their training, sort of learning as they went. Are there ways your people could practice a new technology before critical use? We do have research organizations which try new technologies. They get involved with forward looking projects which may have future impacts. They get the practice in newer technologies. It's difficult to secure funds for practicing.

INSERTED QUESTION II: What do you feel about prototyping? I find prototyping generally targets the easiest parts of large systems first. People tend to gain limited expertise with prototyping and things tend to get bogged down as the major parts are entered.

Closing Comments:

Whatever the reasons are for using a new technology, if it isn't helpful to normal development then it isn't much use.

I don't know if you were told this or not but the faults per lines of code were far higher in the AFCAP project than in our other projects using the older traditional methods.

Pooled Data

This section pools two categories of information revealed in the interviews so that the data can be evaluated as a group. These two categories are user team personality data and education level of all interviewed.

User Team Personality

Table 6.1 in chapter VI reflects the primary answers given when interviewees were asked to judge the user team personality. Some interviewees added other comments during this questioning. These comments are below and, again, are not associated with the above positions due to respondent uneasiness with giving these answers.

We missed some schedules and this affected self-confidence. At times it was very low.

My definition of persistence is sticking with a problem to fix the whole problem. Some fixed part of problem and did the rest fast and dirty.

[On persistence] We worked a lot of overtime and I know this caused 3 or 4 people to leave.

[On persistence] No one chose to leave after working 70 hour weeks for 2 years. [I discovered this conflict with the above answer was due to communication problems.]

Education Level

Each interviewee was asked their highest attained level of education. Below are the number and type of degrees found.

Six - Master of Science, Computer Science

Two - Master of Science, Electrical Engineering

One - Master of Business Administration.

One - Bachelor of Science, Computer Science

One - Bachelor of Science, Computer Science and Electrical Engineering

APPENDIX C

PROJECT DETAILS MAILED TO FIRMS

RESEARCH PROJECT ON MANAGING EMERGING TECHNOLOGIES:
OBJECT ORIENTED SOFTWARE DEVELOPMENT
CASE STUDY

PROJECT DETAILS

Research Needs. Academic literature suggests methods to implement object-oriented software development (OOSD) in corporations. However, there are few detailed studies available which support or refute this literature. What is needed is an opportunity to study actual implementations of OOSD so that the literature may be substantiated or that new information may be generated which reflects real-world requirements and practices. A study of your firm's implementation of OOSD would greatly aid this effort.

OOSD Environment. This project views the ideal object-oriented software development environment as utilizing object-oriented analysis (OOA), object-oriented design (OOD), and object-oriented programming (OOP). However, due to the limited OOA tools and the recent maturing of OOD methodologies, this project recognizes this comprehensive environment is unlikely to be encountered in 1992. Therefore, the minimum environment sought is a firm using OOP and at least a hybrid OOD technique to identify and design the necessary classes and objects. Your participation in this study would be extremely valuable whether you have finished an OOSD system or are currently developing an OOSD system.

Interview Population. The effective evaluation of the management and technical factors impacting OOSD requires interviewing several key people of your firm. We would like to talk to the following people:

a. The person given responsibility for overseeing the successful implementation of OOSD - often referred to in literature as the "project champion."

b. At least two managers (if applicable) who are/were assigned to assist the project champion.

c. Seven software developers who are/were assigned to the same object-oriented system:

- Two analysts (or your equivalent).

- Two designers.

- Two programmers.

- One software maintainer (if one was assigned to evaluate the maintainability of the effort).

d. At least one senior manager who had a part in deciding to implement OOSD.

e. Two support people from your equivalent of the technology resource center. (This organizational unit is sometimes referred to as the information center. Its purpose is to obtain the needed resources and training to implement OOSD.)

Time Period for Data Gathering. The total number of people we would like to interview is 13. The estimated average time required per

interview is 30 minutes. Two to three interviews may be 10-15 minutes less while two to three may be 10-15 minutes longer. The total estimated time to complete all interviews is 6 1/2 hours. The researcher would be at your location for 3 days with total flexibility in interview times. Please feel free to set any interview schedule you would like over these 3 days.

Voluntary Participation. We request all participation in this study be voluntary. Participants may omit answers to any question they wish.

Confidentiality. No participant names will be published in the final report nor will their responses be attributed to their names in other possible research discussions. Your company's name will not be published in the final report. If there are other questions concerning confidentiality, please call.

Benefits of Research. Your firm will receive a post-implementation report (mid-implementation for a current effort). Topics of the report will include management to development team communications, extent of object paradigm use, review of efforts to reap OOSD benefits, resource selection process, and prognosis of OOSD diffusion to other software efforts. Upon request, this report can be tailored to provide other information within the scope of the research parameters.

Work I did while you were gone were as follows:

CLEANED!!!!!!!!!!

The following Ed Plans were processed:
(This includes updating cards, aces, and folders.)

Capt John K. Gay, revised
Capt Melvin E. Allen, initial
Capt Peter D. Read, initial
Capt Scott D. Mattson, initial
Capt Jon K. Wisham, initial
1st Lt Sean M. Farrell, initial
2nd Lt Lowell E. Bailey Jr., initial
Capt David M. Deloach, initial
Capt James M. Child, final
Capt John E. Meskel, initial
Capt Victor J. Valdez, initial
Capt Stanley E. Grant, initial
1st Lt Mike Eliason, initial
1st Lt Randall J. Redell, revised
Capt Judity A. Wiser, initial
Capt James P. McCombe, initial

CLEANED!!!!!!!!!

The following Ed Plans were returned for the following:

Capt Jerald R. Warner, needs advisors signature
1st Lt Curt D. Wagner, incomplete ed plan
2nd Lt Martin F. Payne, incomplete ed plan
1st Lt Catherine M. Morgan, needs advisors signature
2nd Lt Mark W. Babione, incomplete ed plan
Capt Louann J. Woods, incomplete ed plan
1st Lt Katherine A. Germain, incomplete ed plan

CLEANED DRAWERS!!!!!!!!!!!!!

Processed Leave Forms for the following:

Capt Mark E. Kraus, permissive TDY
Capt Gordon Hendrickson, ordinary
Capt Christopher B. Felt, ordinary
Lt Laura Smith, permissive
Capt Paul K. Daly, permissive

CLEANED!!!!!!!!!!!!!!!!

The following TDY requests were processed:

Capt Andrew Dembosky, thesis research
Capt Brian Hoey, revoked

The following were updated in ACES:

Students within 6 mos of graduation w/o a thesis status and/or payment code or where the graduation date is blank (this is in the thesis menu and I attached the list)

All incoming students

All Naval Postgraduate School students

The following letters were mailed requiring forms:

Capt Merrill Adkison
2nd Lt Daniel Allen
Maj Stephen Atkins
2nd Lt Mark Babione
2nd Lt Lowell Bailey
Capt Kevin Baggett
Capt Hugh Bowman
Capt Ronald Cournoyer
2nd Lt Thomas Crimmins
Capt Donald Dishong
Capt Daniel Elmore
Capt David Hamilton
1st Lt Katherine Germain
Capt Stephen Moree
2nd Lt Michael Kayser
Capt Theodore Lewis
Capt Robert Smith
Capt Nickolaus Behner
Capt Stuart O'neill
Capt Peter Read
Capt Robert Miranda
Capt Michael Marzec
1st Lt Julie Hurford
Capt Stanley Grant
Capt Christine Schubert
1st Lt Barbara Bonner

CLEANED COMPUTERS!!!!!!!!!!!!!!

The following training reports were completed and mailed to CBPO:

Capt Kristen A. Dotterway, final
Capt Adam F. Grove, final
Capt Bennett K. Larson, final
Capt David A. Brockway, final
Capt Lee E. Thomas, annual
Capt Mark E. Kraus, annual
Capt Robert L. Schantz, final
Capt Theresa L. Pobst-Martin, final
Capt Stephen G. Cunico, annual
Capt Gary L. Crowder, annual
Capt Brian G. Fillmore, annual

Capt Bruce O. Fagerland, annual
Capt William M. Major, annual
Capt David L. Ritter, annual
Capt James M. McVay, final
Capt James Child, final
Capt Michael A. Cervi, final
Capt Jeffrey A. Ralston, annual
Capt Wayne R. Martin, final

CLEANED DESKS!!!!!!!!!!!!!!!!!!

The following students are closed-out and returned to AFIT/RR but a final ed plan needs to be request to send to CIF:

Capt Theresa L. Pobst-Martin
7450 Tactical Intel SQ (USAFE) new address
Ramstein ABS Germany
APO, AE 09094

Capt Krista Evans
Tactical Air Cm OL ACOO (TAC)
Langley AFB VA 23655-6000

CLEANED FILING CABINETS!!!!!!!!!!!!!!!!

The following students are closed-out and returned to AFIT/RR:

Capt Kristen A. Dotterway
Capt Adam F. Grove
Capt Bennett K. Larson
Capt David A. Brockway
Capt Robert L. Schantz
Capt Theresa L. Pobst-Martin
Capt Donald G. Rose
Capt Evelyne M. Conlon
Capt Krista L. Evans
Capt Michael A. Cervi
Capt James M. Child
Capt Wayne Martin
Capt James McVay